

AEROPHYSICS RESEARCH CORPORATION
TECHNICAL NOTE

JTN-11

(NASA-CR-141598) THE ENGINEERING DESIGN
INTEGRATION (EDIN) SYSTEM (Aerophysics
Research Corp., Houston Tex.) 239 p HC

N75-17118

CSCL 09B

Unclas

G3/61

09633

THE ENGINEERING DESIGN INTEGRATION
(EDIN) SYSTEMby: C. R. Glatt, G. N. Hirsch, G. E. Alford,
W. N. Colquitt and S. J. Reiners

prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Johnson Space Center
Houston Texas 77058

December 1974

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
US Department of Commerce
Springfield, VA. 22151

PRICES SUBJECT TO CHANGE

N O T I C E

**THIS DOCUMENT HAS BEEN REPRODUCED FROM THE
BEST COPY FURNISHED US BY THE SPONSORING
AGENCY. ALTHOUGH IT IS RECOGNIZED THAT CER-
TAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RE-
LEASED IN THE INTEREST OF MAKING AVAILABLE
AS MUCH INFORMATION AS POSSIBLE.**

1. Report No. NASA CR-		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle THE ENGINEERING DESIGN INTEGRATION (EDIN) SYSTEM.				5. Report Date	
				6. Performing Organization Code	
7. Author(s) C. R. Glatt, G. N. Hirsch, G. E. Alford, W. N. Colquitt and S. J. Reiners				8. Performing Organization Report No.	
9. Performing Organization Name and Address Aerophysics Research Corporation 18100 Nassau Bay Drive, #147 Houston, Texas 77058				10. Work Unit No.	
				11. Contract or Grant No. NAS9-13584	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Lyndon B. Johnson Space Center				13. Type of Report and Period Covered Contractor Report	
				14. Sponsoring Agency Code	
15. Supplementary Notes					
16. Abstract The report provides a description of the Engineering Design Integration (EDIN) System as it exists at Johnson Space Center. A discussion of the EDIN System capabilities and applications are presented.					
<div style="text-align: center; font-weight: bold; font-size: 1.2em;">PRICES SUBJECT TO CHANGE</div>					
17. Key Words (Suggested by Author(s)) Design, synthesis, executive, EDIN, data base, design simulation, multiple programming.			18. Distribution Statement Unclassified - Unlimited		
19. Security Classif. (of this report) Unclassified		20. Security Classif. (of this page) Unclassified		21.	

AEROPHYSICS RESEARCH CORPORATION
TECHNICAL NOTE

JTN-11

THE ENGINEERING DESIGN INTEGRATION
(EDIN) SYSTEM

by: C. R. Glatt, G. N. Hirsch, G. E. Alford, .
W. N. Colquitt and S. J. Reiners

prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Johnson Space Center
Houston Texas 77058

December 1974

PREFACE

This report describes the Engineering Design Integration (EDIN) system in operation at Johnson Space Center (JSC) at the completion of the contract NAS9-13584, "Extended Optimal Design Integration (Extended ODIN) Computer Program." The study was carried out in the period from June 1973, through December 1974, with funds provided by the National Aeronautics and Space Administration, Johnson Space Center, Engineering Analysis Division, Launch Analysis Section. The report specifically describes the Univac 1110, Exec 8 EDIN system at JSC. Earlier systems adapted to the CDC 6600 have been installed at the NASA Ames Research Center facility, the USAF Flight Dynamics Laboratory facility and the Langley Research Center facility.

The original ODIN system was initially developed during 1971 and 1972 under parallel contracts with the National Aeronautics and Space Administration, LRC (NAS1-10692) and the USAF Flight Dynamics Laboratory (F33615-71-C-1480). The original system was developed for CDC 6000 series computers but the system has since been converted to the Univac 1100 series computer under contract (NAS9-12829) to NASA Johnson Space Center.

The present contract (NAS9-13584) resulted in the expansion of the EDIN technology library, the development of a number of general and special purpose utilities, the development of a geometry technology module and an executive data processor. Numerous applications of the software developed have been accomplished, some of which are presented herein.

TABLE OF CONTENTS

	Page
SUMMARY.....	1
INTRODUCTION.....	3
Study Objectives.....	4
Study Approach.....	4
THE EDIN SYSTEM.....	6
THE COMPUTER.....	6
Computer Operating System.....	6
The Executive System (Exec 8).....	8
System Processors.....	12
Language Processors.....	13
Utility Processors.....	13
Subroutine Libraries.....	14
Applications Programs.....	15
Computer Concepts and Definitions.....	15
Absolute Element.....	15
Collection.....	15
Element.....	15
Language Processor.....	15
Program.....	16
Processor.....	16
Relocatable Element.....	16
Symbolic Element.....	16
Catalogued File.....	16
Cycle.....	16
Data Base.....	17
Data File.....	17
External File Name.....	17
File.....	17
Internal File Name.....	17
Master File Directory.....	17
Private File.....	17
Program File.....	17
Public File.....	17
Project.....	17
Qualifier.....	17
SDF Format.....	18
Temporary Program File (TPF\$).....	18
Temporary File.....	18
Run Stream Concept.....	18
File Names and Element Names.....	19
Executive Control Statements.....	20
Control Statement Format.....	20
Label Field.....	20
Operation Fields.....	21
Operand Fields.....	21
Transparent Control Statements.....	21

TABLE OF CONTENTS (Continued)

	Page
Summary of Control Statements.....	21
File Utility Routines.....	22
Systems Symbolic Processors.....	26
ELT Processor.....	26
Data Processor.....	26
ED Processor.....	27
CULL Processor.....	27
LIST Processor.....	27
The Collector.....	29
DATA MANAGEMENT SYSTEM.....	31
DMAN Software Package.....	33
DMAN Usage.....	34
A Discussion of IT.....	35
A Discussion of IOP.....	35
The DLG Processor.....	37
DLG Usage.....	38
Control Statement.....	38
Option Specifications.....	38
Syntax Definition.....	40
Summary of DLG Directives.....	40
Descriptions of Control Directives.....	41
Processor Interface.....	45
Usage.....	45
Restrictions.....	46
Technology Module Interface Package.....	47
PROGRAM LIBRARY.....	50
Geometry.....	50
Aerodynamics, Stability and Control.....	50
Propulsion.....	51
Mass and Volumetric Properties.....	51
Performance.....	52
Thermodynamics.....	52
Structures.....	53
Cost.....	53
Environmental Protection.....	53
ABLATOR: One Dimensional Analysis of the Transient Response of Thermal Protection Systems.....	54
ACMOTAN: Linear Aircraft Motion Analysis.....	54
AESOP: Automated Engineering and Scientific Optimization Program.....	55
AFSP: Automated Flutter Solution Procedure.....	55
AIRFOIL: A Program for Generating Geometric and Aerodynamic Characteristics of Airfoil Sections.....	55
ATOPII: Atmospheric Trajectory Optimization.....	56
COAP: Combat Optimization and Analysis Program.....	56
CONPLOT: Aircraft Configuration Plot.....	57
DAPCA: Development and Production Cost of Aircraft.....	57

TABLE OF CONTENTS (Continued)

	Page
DATCOM: Configuration Design Analysis Program (TRW).....	57
DATCOM2: Configuration Design Analysis Program (MDAC).....	57
ENCYCL: Design Point Performance of Turbojet and Turbofan Engine.....	58
GENENG: A Program for Calculating Design and Off- Design Performance for Turbojet and Turbofan Engines.....	58
GENENGII: A Program for Calculating Design and Off- Design Performance of Two and Three Spool Turbofans with as many as Three Nozzles.....	58
GEOMETRY: Body Coordinate Generator.....	59
HABACP: Hypersonic Arbitrary Body Aerodynamic Computer Program.....	59
HDG: Heading Program.....	59
IMAGE: Configuration Display Program.....	60
MINIVER: Aerodynamic Heating Program.....	60
PADS: Performance Analysis and Design Synthesis Program.....	60
PANEL: A Program for Generating Panelled Con- figuration Geometry.....	60
PLTVIEW: Program for Generating Separation Plots...	61
POST: A Program to Optimize Simulated Trajectories.	61
RDPRO: POST Plot Data Generation Program.....	61
PRESTO: Program for Rapid Earth-to-Space Tra- jectory Optimization.....	62
PRICE: A Program for Improved Cost Estimation.....	62
REPORT: Report Generator.....	62
SBOOM: Sonic Boom Prediction for Shuttle Type Vehicles.....	63
SEPARATE: The Program to Simulate Separating Stages of Launch Vehicles.....	63
SKINF: Turbulance Skin Friction Drag Program.....	63
SSAM: Swept Strip Aeroelastic Model.....	64
SSSP: Space Shuttle Synthesis Program.....	64
TOLAND: Take-Off and Landing Program.....	64
TOP: Trajectory Optimization Program.....	65
TOPLOT: Plot Generator for TOP.....	65
TREND: Subsonic/Supersonic/Hypersonic Aero- dynamic Trade-Off Program.....	65
VAMP: Volume, Area and Mass Properties.....	66
VSAC: Vehicle Synthesis for High Speed Aircraft....	66
WAATS: Weights Analysis for Advanced Trans- portation Systems.....	67
WDRAG: Zero-Lift Wave Drag Program.....	67
WETTED: Wetted Area in Reference Length Program....	67
LIBRARY ADDITIONS AND MODIFICATIONS.....	68

TABLE OF CONTENTS (Continued)

	Page
AIRFOIL: AIRFOIL GENERAT ON PROGRAM.....	70
CIPHER: REPORT WRITING PROGRAM.....	72
Physical Characteristics.....	72
Program Usage.....	73
COT PROCESSOR.....	82
FLOWGEN: AUTOMATIC FLOW CHART GENERATOR PROGRAM.....	83
Program Control.....	83
Program Input.....	83
GTM: GEOMETRY TECHNOLOGY MODULE.....	85
Program Description.....	85
Physical Characteristics.....	89
Program Usage.....	90
Control Cards.....	90
Program Input.....	90
Master Level Language.....	90
Descriptions of the Commands.....	92
Image Input.....	92
Cluster Edit.....	95
Edit Commands.....	95
Output Commands.....	96
Transformation Commands.....	96
Display Commands.....	97
Description of Commands.....	98
Scaling Parameters.....	104
Bounding Commands.....	105
Register Commands.....	106
Miscellaneous Commands.....	106
Segment Edit.....	107
Point Edit Commands.....	107
Segment Level Command.....	111
Program Output.....	112
HABACP: HYPERSONIC ARBITRARY BODY AERODYNAMIC COM- PUTER PROGRAM.....	113
IMAGE DISPLAY COMPUTER PROGRAM.....	116
The Display Device.....	116
Virtual Graphics.....	116
Program Usage.....	117
General Program Information.....	117
\$TYPE32 Inputs.....	117
Geometry Data.....	121
\$TYP343 Inputs.....	121
PANEL: GEOMETRY GENERATION PROGRAM.....	127
Program Description.....	127
COPY5 Module.....	127
ELLIPS Module.....	129
Cubic Patch Module.....	129
TANK Module.....	132
Double Bubble Tanks.....	132

TABLE OF CONTENTS (Continued)

	Page
TZOID Module.....	132
Physical Characteristics.....	132
Program Usage.....	136
Program Output.....	141
DBBOUT Output Descriptions.....	141
TANK Output Descriptions.....	143
PLOTTR: AN INDEPENDENT COMPUTER PROGRAM FOR THE GENERATION OF GRAPHICAL DISPLAYS.....	144
Program Description.....	144
Plot Data Input Option.....	146
Array Format.....	146
Observation Format.....	146
Alternate Data File.....	147
X-Y Plots.....	147
Plot Positioning.....	149
Line Type and Symbols.....	150
Data Scaling.....	150
Scale Annotation.....	151
Grid Generation.....	151
Title Generation.....	151
Auxiliary Plot Text.....	151
Virtual and Display Window.....	152
The Virtual Window.....	152
Contour Plots.....	155
Generation of Contour Vectors.....	155
Two Ambiguous Cases.....	159
Loading the Data.....	160
Contour Definition.....	161
Title and Axis.....	161
Program Usage.....	162
Program Input.....	163
\$PLOTIN Namelist Data.....	164
Physical Characteristics.....	170
Program Loading.....	170
RDPRO: POST PROFILE TAPE READ PROGRAM.....	174
Program Usage.....	174
Example Input.....	174
Program Output.....	175
SFIT: A FURFACE FITTING PROGRAM.....	178
Program Description.....	180
Program Usage.....	181
Control Cards.....	181
Program Input.....	181
\$IN Inputs.....	184
\$SECT Inputs.....	184
Program Flow Logic.....	184
Program Output.....	184

TABLE OF CONTENTS (Continued)

	Page
SIZER: A PRELIMINARY SIZING PROGRAM FOR LAUNCH VEHICLES	186
Program Description.....	186
Program Usage.....	188
Input Procedure.....	188
Flow Logic.....	188
Program Output.....	188
TANK: A PRELIMINARY TANK DESIGN PROGRAM.....	192
Program Description.....	192
Physical Characteristics.....	192
Program Usage.....	194
Program Input.....	194
Program Output.....	196
VL70: A PROGRAM FOR READING AERODYNAMIC DATA TAPES.....	197
Program Description.....	197
Physical Characteristics.....	199
Program Usage.....	199
Program Input.....	199
Program Output.....	199
WAB: A PROGRAM FOR COMPUTING WEIGHTS AND BALANCES.....	202
Program Description.....	202
Program Usage.....	203
Control Cards.....	203
Input Procedure.....	203
\$IN Input Set.....	204
\$BOX INPUT SET.....	205
Fixed Format Surface Data.....	205
Program Outputs.....	206
Program Flow.....	206
APPLICATION OF THE SYSTEM.....	209
Heavy Lift Booster (4 SRMS).....	209
Heavy Lift Booster (5 SRMS).....	209
Nuclear Waste Disposal Heavy Lift Booster.....	209
SOL47B Payload/Body Volumetric Study.....	211
SOL47B Heavy Lift Booster.....	211
Space Station Evaluations.....	211
Evaluations of Shuttle with the J2S Engine.....	211
Shuttle Orbiter c.g. Analysis.....	212
Sample Analysis.....	212
Definition of Analysis Tasks.....	212
Selection of Analysis Programs.....	214
Intercommunication Data.....	214
Generation of a Run Stream.....	215
CONCLUDING REMARKS.....	225
REFERENCES.....	226

THE ENGINEERING DESIGN INTEGRATION (EDIN) SYSTEM

By: C. R. Glatt, G. N. Hirsch, G. E. Alford,
W. N. Colquitt and S. J. Reiners

Aerophysics Research Corporation

SUMMARY

The EDIN system is a digital computer program complex for the evaluation of aerospace vehicle preliminary designs. The system consists of a Univac 1100 series computer and peripherals using the Exec 8 operating system, a set of demand access terminals of both the alphanumeric and graphics types and a library of independent computer programs. The program library contains programs for estimating all major flight vehicle characteristics such as aerodynamics, propulsion, mass properties, trajectory and mission analysis, cost, steady state aeroelasticity, flutter and stability and control. There are also utility programs in the library for generating and controlling the flow of design data within the computer program complex.

The data base used by the EDIN system consists of dynamically constructed name addressable region of online storage which can be subdivided into user established classes of data. Each class represents a subset of the available data. A data manipulation program called the DLG processor is used for the construction and maintenance of the data base. It interrogates the data base to satisfy data interface requirements among the technology programs. In addition, the DLG processor can address all or portions of online structured data sets for insertion as part of the input stream of the technology programs.

The executive control of library program execution is performed by the Univac Exec 8 operating system through a user established run stream. Any set of vehicle component matching and sizing loops can be defined by partial run streams (or design sequences) consisting of control cards and/or technology input data. There is no effective limit on the number of design sequences which may be established by the partial run stream technique. A data linkage of the design sequences with the data base is provided by the DLG processor. DLG modifies the partial run streams in accordance with user instructions within the partial run stream.

The EDIN system can be operated in a demand or batch environment. In the demand mode some interaction with the data base and technology modules is available. Usually a combination of

demand and batch operations is employed in the evaluation of preliminary designs using this sytem. Several applications representing samples of the type of design support activity have been accomplished with the EDIN system.

These applications along with a description of new and existing software in the EDIN system are as described in this report.

INTRODUCTION

A great deal of effort in the last several years has been expended in the pursuit of computer aided design capability which can be used effectively in the engineering design environment. These efforts have met varying degrees of success. References 1 through 16 describe a number of approaches to the problem. Most capabilities have been developed to support design synthesis for the study of vehicle concepts. They are usually implemented by one of two methods:

1. The vehicle under study is completely synthesized within a single computer program.
2. The vehicle under study is analyzed using separate technology modules which are linked by independent executive program control.

The use of the first method requires extensive computer programming and checkout before simulations can begin. The method is characterized by very large programs, limited applicability and complex data setup requirements. Programming bugs that appear after program checkout can cast doubt on simulation results. In addition, program modification required for design concept changes can cause delays in the engineering analysis process. In summary, the single program concept does not provide for a powerful computer aided design capability.

The use of the second method eliminates the need for large programming efforts allowing design simulations to begin almost immediately after a concept is formulated. Data setup is similar to the setup of the individual TM's. The executive program control concept effectively supports design simulations for conceptual studies involving a relatively small number of users. The concept does not provide the computer aided design environment necessary to support a large design staff nor does it permit sufficient user interaction with the analysis process.

Study efforts at JSC during the past eighteen months have resulted in the development of the EDIN (Engineering Design Integration) System. The system uses the independent technology module (TM) concept of reference 9 and an expanded version of the data management system of reference 10. The Univac Exec 8 operating system provides the program linking capabilities and the file management software to drive the EDIN system. A data processor called DLG has been developed for merging data base information with the normal input stream of the TM's and for storing technology data from the TM's in the data base. An interactive geometry module, which also uses the EDIN data base concept, has been developed for the generation, maintenance and storage

of geometry data. The stored geometry can be retrieved and used by other technologies. Both two and three dimensional graphical analysis modules have been adapted for use in the EDIN system and a number of general and special purpose utilities have been developed to support design analysis.

The EDIN system, as it currently exists, can be operated in a batch or demand mode. The system has limited interactive capability but lacks the essential hardware features to qualify it as an interactive system. A typical operating mode, is a combination of demand and batch modes of operation. The user of the system constructs a run stream from stored input data elements. The run stream consists of design sequences (partial run streams) of technology modules selected for the design analysis at hand. The most current design information is obtained from the EDIN data base. A configuration analysis is performed using the Geometry Technology Module and auxiliary programs. The constructed run stream is submitted to the computer for execution and a design analysis follows. The user has options of updating the data base with the information obtained and/or generating summary report information to support his analysis.

Study Objectives

The objectives in the development of the EDIN system are to provide Computer Aided Design (CAD) capabilities, which span the engineering functions of the preliminary design process to reduce the design analysis time and to improve the integrity of the design. To support the objectives, JSC has provided to the EDIN project storage tube terminals, teletype compatible terminals and a MOPS terminal connected to the Univac 1100 series host computer. The terminals provide the primary interaction with the large scale TM's executing on the host computer. The EDIN project staff was charged with the responsibility of software development required to support the objectives.

Study Approach

The study approach placed significant emphasis on the application of the developing system. The NASA technical monitor imposed study requirements on the EDIN project. These requirements were used by the EDIN project staff to perform the required design analysis. The staff members defined specific program developments using the contract statement of work as a guideline. The development phase was aimed at the modification or development of software which more adequately supported the original design analysis. Following the program development phase, the same or similar design analysis was performed using the newly developed software. The sequence was repeated for each set of NASA directed study requirements.

The study approach is considered to have provided the best possible end product for the funds expended. This volume presents the state-of-the-art of the EDIN system at JSC and describes the software developments and applications performed under the contract.

THE EDIN SYSTEM

The components of the system, shown schematically in figure 1, consist of the computer and a library of the technology modules (TM), utilities and data bases. The system can be operated from demand terminals in a batch mode or a combination thereof. All of the major technologies, such as aerodynamics, propulsion and structures, are represented in the TM library. The utility elements include graphics, plotting and report generation. The technology programs provide the real design analysis capability which must be performed to obtain evaluation of any vehicle design. The utility elements aid the designer in obtaining a better understanding of the results of the analysis and provide a means of improving or expediting the design analysis. They also permit the designer to transfer or transform data generated by one program for use by other programs or for use by the analyst.

Since the system comprises literally millions of source cards, some precautions have been taken to provide a usable system capable of interpretation by designer, engineer and programmer. The major precaution has been the creation of a system which is truly modular in the sense that it consists of many independent computer programs. Any one of these programs can be revised, extended or replaced without affecting the other program elements of the system in any way. External to the technology programs themselves, a data base of common information is maintained by an executive data processor. Each technology program may draw upon the data base for information as required. As a consequence of the construction concepts employed, the specialists in any technology area are able to phrase the analysis of the design without regard for the other technologies involved other than the interfaces with the common data base. The common data base attributes are defined by the design staff and can consist of all information which is communicated between elements or communicated from the EDIN system itself to the staff. The data represents a subset of the total amount of information generated by all of the programs within a given design sequence. When combined with the normal input data, it is sufficient to completely define the program under study. When combined with the normal output, it represents the data description of the vehicle performance and mission requirements.

THE COMPUTER

Computer Operating System

The computer operating system selected for EDIN is the Univac supplied software for the 1100 series computer including the executive system (Exec 8), compilers, utilities, subroutine libraries, etc. The EDIN system draws heavily from the executive

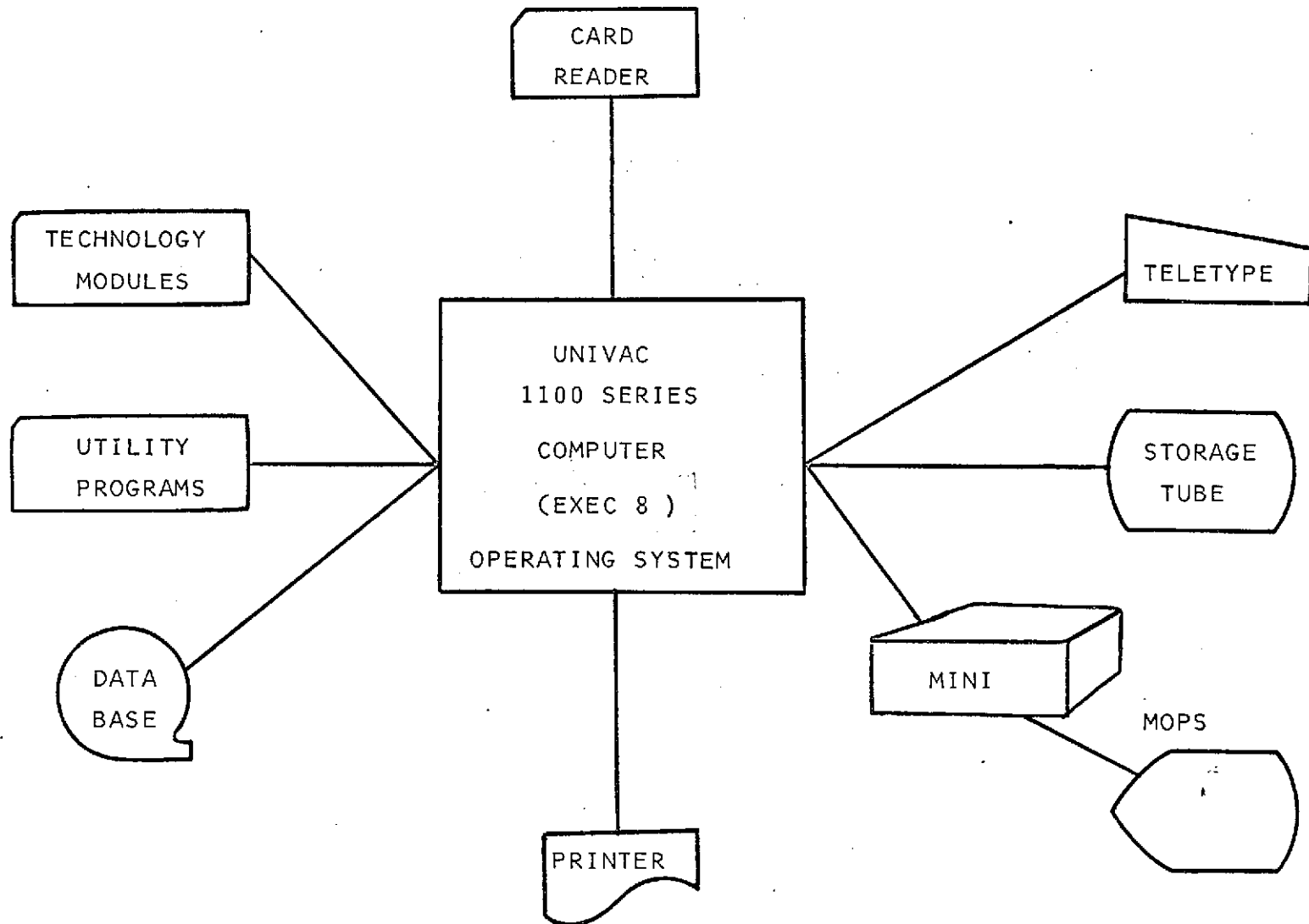


FIGURE 1

COMPONENTS OF THE EDIN SYSTEM.

functions for program control and file management. A knowledge of the control functions of Exec 8 are essential to the user of the EDIN system. Those control functions which have the greatest applicability to design analysis are offered to the potential EDIN user in this section. Detailed information is provided in the Univac manuals listed in figure 2.

The Univac 1100 operating system is an outgrowth of Univac's many years of experience in multiprogramming, multiprocessing, time sharing, communication and real time orientation systems. It provides the flexible user environment which is essential to the operation of the EDIN system. A complete set of software ranging from high level language compilers to basic service functions is included in the operating system. The six major categories are:

1. Executive System, (Exec 8).
2. System Processors.
3. Language Processors.
4. Utility Processors.
5. Subroutine Library.
6. Applications Programs.

The first three categories represent the base compliment of software supplied with the 1100 series computer and maintained by Univac. They are generally used by the EDIN system without modification. The user portion of the operating system is described in the last three categories. The EDIN system takes advantage of the software which is already available and augments the software capability in these categories with special modules dealing primarily with engineering and design integration. The EDIN interface to the Univac 1100 series computer is shown in figure 3.

The Executive System (Exec 8). - To take full advantage of the speed and hardware capabilities of the 1100 series sytems, a comprehensive internal operating environment is provided in the Exec 8. This environment permits the concurrent operation of many programs and directs the computer to react immediately to the inquiries, requests and demands of many different users at local and remote stations. The Exec 8 system can store, retrieve and protect large blocks of data and makes the best use of available file space.

Only through central control of all activities of the system can this environment of the combined hardware and software systems

UNIVAC 1100 SERIES OPERATING SYSTEM PROGRAMMER
REFERENCE UP-4144 REVISION 3

UNIVAC 1100 SERIES EXEC 8 HARDWARE/SOFTWARE SUMMARY
REFERENCE UP-7824 REVISION 1

UNIVAC 1100 SERIES EXEC 8 FORTRAN V
REFERENCE UP-4060 REVISION 2

UNIVAC 1100 SERIES FORTRAN V LIBRARY PROGRAMMER
REFERENCE UP-7876

NASA INSTITUTIONAL DATA SYSTEMS PROCEDURES MANUAL
PART 11 GRAPHICS LIBRARY
PART 20 EXEC 8 SYSTEM
PART 21 EXEC 8 DEMAND SYSTEM

FIGURE 2 UNIVAC 1100 SERIES COMPUTER SYSTEM REFERENCE MANUALS

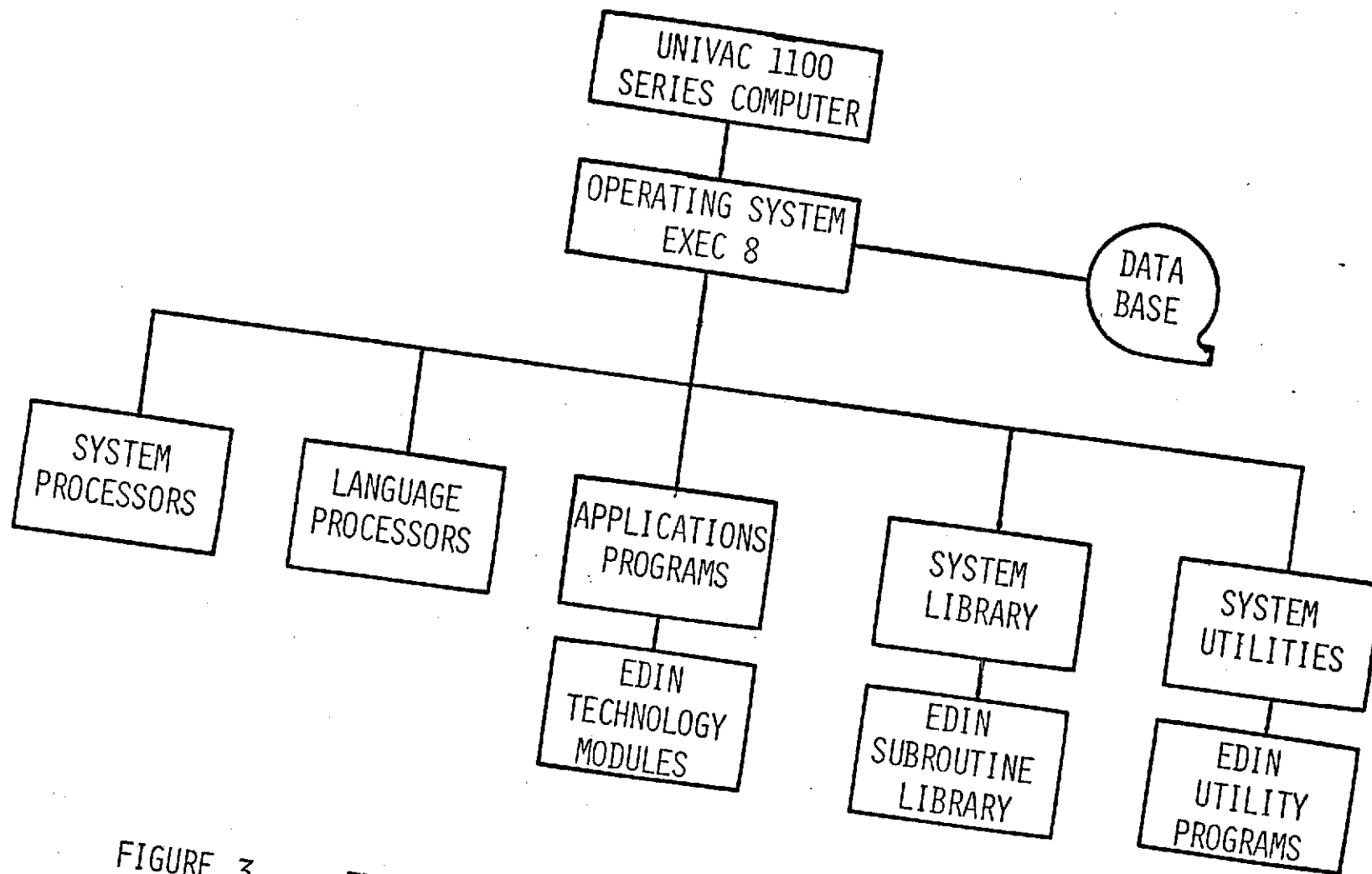


FIGURE 3

EDIN INTERFACE TO THE UNIVAC 1100 SERIES COMPUTER.

be fully established and maintained to satisfy the EDIN requirements. No user level executive can exercise the necessary control. The responsibility for centralized control is borne by the Exec 8 system, which controls and coordinates the functions of the internal environment. A relatively simple interface to the executive, based on the run stream concept, is provided which relieves the users of concern for the internal interaction between his program and other coexistent programs.

The technical capabilities of the Executive System cover a variety of data processing activities. The executive system design offers the versatility of handling batch processing, demand processing and real time processing using multiprogramming and multiprocessing techniques.

Batch jobs may be submitted from remote terminals as well as from central site equipment. The batch mode is used in EDIN applications for long running analysis and programs with high core requirements.

Complementing the batch processing capabilities are the Exec 8 time sharing capabilities. This mode of operation accommodates simultaneous requests and demands from users at numerous remote inquiry terminals operating in a demand (or conversational) mode. All facilities available to the batch processing user are also available in a demand mode. The primary difference being that the demand mode permits the user additional flexibility in the statement and control of individual tasks. For example, when an error is made in the demand mode, the user simply corrects it online and proceeds rather than suffering the turn-around cycle inherent in batch processing. The demand user may interact with the system at various levels which include the Executive System, a conversational processor or with a user program. Many conversational EDIN programs take full advantage of the demand mode. Design analysis can be interrupted at any point to assure integrity using the demand mode.

The executive system is also designed to be applicable to programs which have real time requirements. The Univac Communications Subsystems, together with the scheduling and interrupt processing features of the Executive System provide an environment satisfactory for the operation of real time programs. However, the EDIN system currently uses no real time processing.

The Executive System is designed to ensure effective and efficient utilization of the mass storage devices. The consequence is an unprecedented ability to relieve users of the responsibility of maintaining and handling cards and magnetic tapes, thus eliminating many errors which heretofore have accompanied the

use of large scale software systems. At the same time, the overall operating efficiency is considerably improved. Card handling is virtually eliminated in the use of the EDIN system.

Permanent data files and program files are maintained on the mass storage devices, with full facilities for modification and manipulation of these files. Security measures are established by the Executive System to ensure that files are not subject to unauthorized use. Provisions are also made within the Executive System for automatic relocation of infrequently used files to magnetic tape, as unused mass storage space approaches exhaustion. When the use of files relocated in such a manner is requested, they are retrieved and restored under control of the Executive System with no inconvenience to the user. EDIN makes heavy use of the permanent files storage for the maintenance of technology modules, utilities and data bases.

In the executive system the user has a simple means of directing the execution of the individual tasks of a run and of relaying operational information concerning the run to the executive. This is accomplished through a set of control statements recognized by the executive system. The control language is open ended and easily expanded so that features and functions may be added as the need arises.

The basic format of a control statement is quite simple, and is adaptable to a large number of control functions. Each control statement consists of a heading character @, or in special cases a @@ for recognition purposes, followed by a command and a variable number of parameters. The end of a control statement is indicated by the end of a card, a carriage return or an equivalent signal, depending on the type of input device. The Exec 8 control statements commonly used by EDIN will be discussed.

System Processors. - The system processors are computer programs which provide the functions required to construct and modify other computer programs, maintain and modify data files, and provide diagnostic information upon program termination. The most commonly used system processors are:

The Collector is designed to provide the user with the means of collecting and linking subroutines and to produce an absolute program in a form ready for execution under control of the Executive System.

A File Utility Processor (FURPUR) consists of a set of file maintenance routines which provide the flexibility in management and manipulation of catalogued or temporary files containing data or programs.

The Postmortem Dump Processor (PMD) produces edited dumps of the contents of main storage at program termination. Dumps produced dynamically during execution are automatically printed. Individual program parts are identified with the assistance of diagnostic tables produced with the absolute program by the collector.

The ELT Processor is used to introduce an element into a particular program file or make corrections to a symbolic element in a program file from the run stream.

A File Administration Processor (SECURE) uses a source language structure which allows the user to periodically define specific file administration code tasks. The processor's functions are to produce backup copies of catalogued files, and to provide a recovery mechanism for these files in case of system failure.

The ED Processor is a text editor which enables a user to modify or move character strings in either program files or data files. This program is the primary mode of interaction with the EDIN system program and data files.

The Procedure Definition Processor (PDP) accepts source language statements defining assembler, COBOL, or Fortran procedures and builds an element in the user-defined program file. These procedures may be referenced subsequently in an assembly or compilation with definition. The processor is used in the construction of EDIN programs.

Language Processors. - The operating system provides several language processors, such as Fortran, COBOL, ALGOL and the Assembler. Certain of these processors are specifically designed for demand mode operations. Individual documentation of these processors is provided by Univac.

Utility Processors. - The Utility Processors provide features not essential to the operation of the Exec 8 system but aid the user in the preparation of data, run streams and documentation, Univac supplied utility processors are described below:

FLUSH (Flowcharting Language for User's Simplified Handling) is a processor which accepts Assembler or Fortran format input to produce a flowchart.

The SSG Processor is a general-purpose symbolic stream generator. Any variety of symbolic streams, varying from a file to data to a run stream which configures an executive stream, may be generated. Directions and models for

building of the desired stream images are conveyed to SSG through a skeleton which is written in SYMSTREAM, an extensive manipulative language.

CULL is a processor which produces an alphabetically sorted, cross-referenced listing of all symbols in a specified set of symbolic elements. Provisions are included, via options, to selectively include or exclude defined symbols or symbol groups from the output.

The DOC Processor accepts file or card input and composes it into document format according to the user's specifications. Control statements provide listing and text control, including pagination, justification, indentation and hyphenation. Document maintenance is provided on a line-image basis and by content addressing of text character strings.

LIST is a special-purpose processor that provides edited element listings which include associated element control information not normally of interest to the user. It is intended for debugging of software which deals with program files.

The development of the EDIN system has added many processors in the utility category which will be discussed later.

Subroutine Libraries. - An extensive library of subroutines is provided with the operating system. Subroutines referenced by user programs are automatically included when the absolute program is constructed by the collector. The library elements included fall into the following general categories:

Subroutines that support higher level languages (COBOL, Fortran, ALGOL, etc.).

Subroutines that provide processor interfaces.

Diagnostic Subroutines.

Subroutines for improving the efficiency of file handling.

Service routines, for editing, conversion, segment loading and so forth.

MATH-PACK mathematical functions.

STAT-PACK statistical functions.

The EDIN system contains a growing library of subroutines to aid the user in the construction of design analysis programs and to interface these programs with the EDIN data base.

Applications Programs. The operating system provides many applications programs such as APT, GPSS and PERT. These are described fully in their corresponding manuals. The EDIN technology module library falls in this category of programs. Modules for the evaluation of all categories of engineering analysis are available.

Computer Concepts and Definitions

The Exec 8 operating system is the principal interface between the user of the EDIN system and the computer. Exec 8 is responsible for such functions as time and space allocation of system resources, input-output control, interrupt and file protection. This section describes the computer concepts which are germane to the use of the EDIN system and provides definitions that will prove helpful in understanding the remainder of the documentation.

Absolute Element. - An absolute element is a complete program in binary form suitable for execution by the executive system. Such elements normally occur as output from a collection of relocatable elements with all necessary linkages to external subroutines already defined. The absolute element is the form which all EDIN programs are stored in the program library.

Collection. - A process by which individual relocatable elements are combined to form a complete program (absolute element). The process begins with explicate specification of program elements to be included and typically involves searching subroutine libraries for additional unspecified elements required to satisfy requests from the program under construction. The missing elements are commonly obtained from libraries of relocable subroutines specified by the user. Univac provides an excellent collector called the MAP processor for constructing computer programs.

Element. - An element is a named group of information typically manipulated as a logical subdivision of a file. It often defines a program part such as a subroutine; but can also define groups of Exec 8 control cards and/or input data information to the EDIN technology module. There are three basic types of elements, symbolic, relocatable and absolute. Symbolic elements contain character strings representing source code, data and control statements, which can be read by the user if listed. Relocatable elements are written in binary format and are usually the result of a compilation or as an option to collection via the MAP. The absolute element is a complete program unit as described above.

Language Processor. - The language processor is a computer program whose principal functions include compiling, assembling,

translating or related operations for a specific programming language (for example, COBOL, Fortran, Assembler, etc.). A language processor translates user supplied symbolic information into a machine language which can be processed by the Executive System. The EDIN system uses the Univac supplied language processors but contains no language processors in its library.

Program. - A program is generally a series of instructions in a form acceptable to the computer to be executed as a task. The program usually consists of a collection of subroutines supplied by the user or obtained from alternate library sources collected in the form of an absolute element. The EDIN system is based on the independent program concept and is largely independent programs constructed for the performance of tasks related to engineering design integration.

Processor. - A processor is a computer program that is incorporated as an integral part of the operating system or which has the ability to manipulate files and elements under control of the executive. System processors typically reside on the system library and are evoked in a standardized manner, but are otherwise treated as ordinary user programs. Processors can also be constructed by users for specialized manipulation of system elements and files. The EDIN library contains a number of such user processors to facilitate the transfer of information within the system.

Relocatable Element. - A relocatable element is an element containing a program part such as a subroutine in binary format suitable for combination with other relocatable elements in the construction of an absolute program. Such elements occur most commonly as an output of a language processor.

Symbolic Element. - An element containing information in a format which can be read by a user (typically card images). One common usage of the symbolic element is as source language input to language processors. However, the EDIN system makes extensive use of symbolic elements in the construction of design sequences and for the storage of input data to the technology modules.

Catalogued File. - A catalogued file is a file maintained by the executive for an indefinite period, not necessarily related to the life of a particular run. These files are generally retrievable by runs other than the run which originally created the file. In some cases the catalogued file may be assessable simultaneously by two or more runs.

Cycle. - A cycle is the number used to identify successive updates of files or symbolic elements.

Data Base. - The data base is an element or file of information created for the purpose of storing information for later use. The EDIN data base consists of program files, data files and control statement files used for the analysis of aerospace vehicle designs.

Data File. - A data file is a file in system data format (SDF) created or updated by one of several operating systems mechanisms. Usually a system data processor called DATA is used but the ED processor can also be used.

External File Name. - The external file name is the full name by which a file is defined to the system for cataloguing purposes. In addition to the basic name, full identification requires qualifier and cycle information.

File. - A file is an organized collection of data treated as a unit and stored in such a manner to facilitate access and retrieval of the information.

Internal File Name. - An internal file name is an abbreviated file name used in an individual run stream and related operations concerning a particular file. An internal file name may have an implicit association with an external file name; or it may be associated to a particular external file name by an explicit executive control statement.

Master File Directory. - The master file directory is a directory of file names maintained by the executive to control the retrieval and retention of catalogued information.

Private File. - A private file is a catalogued file that can be assigned and accessed only by runs of a particular project.

Program File. - A program file is a specially structured file containing a group of elements and residing in online mass storage. The program file can consist of symbolic, relocatable or absolute elements.

Public File. - A public file is a catalogued file that can be assigned and accessed by a run of any project.

Project. - A project is an accounting identification for a user of computer resources.

Qualifier. - A qualifier is an extension of the basic name of a file which can be employed by the user to resolve a variety of ambiguous naming situations. Every file has a qualifier which is normally applied according to executive system conventions other than being specifically stated in references to the file.

SDF Format. - The system data file format is the standard data format employed by the operating system. SDF is a sequential fixed block, variable record format in which records may span more than one block of information.

Temporary Program File (TPF\$). - A mass storage file assigned automatically by the executive to each run as a convenience to the user in a wide variety of program file and element manipulation operations. The file is assumed as a program file in the absence of an explicit file name reference.

Temporary File. - A translucent file created by and accessible to and existing within the life of a single run only. Temporary files are assigned to the run stream for the purpose of temporarily storing information which will not be needed at a later date.

Run Stream Concept

The run stream concept is employed as the primary interface between the computer operation system and the user. The run stream contains the specification of the tasks which will be performed by the computer. It is the largest working group read and manipulated by the executive system. The run stream itself is a sequence of data images which taken as a whole constitute the specifications of a run. The run stream consists of an @RUN control statement followed by other control statements and data which direct the performance of individual tasks. Each task consists of one or more control statements which provide technology oriented or utility operations. Tasks may also be grouped into partial run streams. The partial run stream may be added at any point in the run stream by use of the @ADD control statement. The run stream is terminated by a @FIN card which directs the executive to terminate processing of the run stream. In the batch stream of operation, the entire run stream is normally stored on the mass storage facilities before run processing is initiated. The executive system schedules the run stream as a unit. Once initiated, the executive processes the entire run stream. In a demand mode, the run is normally initiated immediately upon acceptance of the @RUN control statement. The system continually solicits the demand terminal for additional run stream input which generally occurs dynamically or on an interactive or conversational basis. The demand solicitation continues until an @FIN control statement is submitted from the demand terminal. Once the run is opened, the executive processes the control statements as they are encountered. A batch run terminates as the result of an abnormal task termination. However, in the demand mode, an abnormal task termination will not

terminate the run but simply print the diagnostic message which causes the abnormal task termination and solicits another executive control statement.

Every run has an output print file assigned to it by the executive which is defined at run initiation. However, the print file may be reassigned during execution of the run stream. In general, all control statements, diagnostic messages and summary accounting information are printed on the print file as well as primary output from the user programs. An output punch file is also created if any user tasks generated punched card output. In the batch mode all output files are printed at run termination on a line printer, at the site from which the run was initiated. In the demand mode printed output occurs at the terminal as it is generated. Punched output occurs at the central site. Printed output may be directed to alternate files through the use of executive control statements. The alternate print files can be temporary files which are deleted at the end of the run or they can be permanent files to be edited or printed at a later date. The alternate print file capability is the primary method of controlling the output from large EDIN design simulations. Print files are selectively created and distributed in accordance with design analysis requirements.

File Names and Element Names

Each file in the operating system is assigned an unique name to distinguish it from all other files. The file name is required in many control statements and directives that are used to reference files. The following notation is used to specify file names:

QUALIFIER*FILE NAME(F-CYCLE)/READ KEY/WRITE KEY.

ELEMENT NAME/VERSION(E-CYCLE)

File name is used with the basic name of the file and qualifier is used to establish uniqueness between files that have the same basic name. F-Cycle is used to identify a particular file from a set of related files that have the same qualifier and file name. Read key and write key are used to obtain read and write access respectively to the file. These keys are not a part of the file name for purposes of assigning a file.

Qualifier and file name each consist of from one to twelve characters selected from the set A to Z, 0 to 9, minus and dollar. F-Cycle is an integer number. Read key and write key each consist of one to six characters. Any character may be used except blank, comma, slash, period and semicolon. File names and element names are used extensively in the EDIN system since the

data base elements are constructed within the Univac 1100 series mass storage media.

Executive Control Statements

Control of the operating system for the Univac 1100 series computer is accomplished through a set of executive control statements. These statements direct the executive system in the processing of a run. Control statements may evoke executive functions such as scheduling, assignment of facilities (files) etc. or may cause the execution of a user program or a processor (that is, a task). The executive control statements are designed in a compact and descriptive manner to facilitate use and yet provide access to all of the features of the executive system. A complete description of the executive control statements is provided in the Univac operating system manuals (see figure 2). A summary is provided here to familiarize the potential user of the EDIN system with the background necessary to understand how the EDIN system operates.

Control Statement Format. - Control statements consist of the recognition character @ and three major sections:

The Label Field

The Operation Field

The Operand Field

A transparent control statement format is also provided that consists of a double recognition character @@ followed by only the operation and operand fields. The transparent control field is used during the execution of a program, processor or task. Each of the control sections on the control statement may contain one or more parameter fields and each of the parameter fields may be further subdivided into parameter subfields. The recognition character is called a master space @ on a terminal or a 7/8 punch for punched cards or equivalent for other devices. The recognition character must always appear in column 1. The format of the control statements with certain exceptions is free form. That is the order of the parameters fields within the control statements is fixed but the parameter fields are not restricted to a particular column.

Label Field. - The label field is optional on the control statement and is generally used to identify a position in the run stream to which control is to be skipped following a dynamic adjustment to the run stream. The label is used in the EDIN system for controlling design sizing and matching loops.

Operation Fields. - The operation fields contain the command and options associated with the command. Unless the control statement is used only as a label statement, the command field must always be specified as it determines the basic operation of the control statements. The command actually specifies the execution of a system processor and the options are control characters which are interpreted by that processor to specify the actions which the processor will perform. The command field is terminated by one or more blanks, or if options are specified, by a comma. On some control statements the options may be broken into subfields, each of which is separated by a slash (/). A blank character or series of blank characters separate the operation field from the operand fields.

Operand Fields. - The operand fields specify parameters associated with the command fields. They are separated by commas and are specified by the user as dictated by his requirements. The content of the operand field, the number of operand fields and whether each is required or optional varies with the command selected. Operand fields in turn may contain parameter subfields that are separated by the slash. For the most part, these subfields are optional within a field. Thus it is possible to specify parts of a field without specifying the entire field. A common use of the operand field is to specify file names and element names on which the control statement is applied. However, many other uses are made of these fields by the system processors.

Transparent Control Statements. - A special mode of processing control statements is available during demand processing. This mode directs the executive to process the control statement immediately after it is input from a remote entry terminal. The processing called for by the control statement is also done independently of any current program execution or control statement processing in the run stream. This mode of executing a control statement is specified by a double master space @@ on the control statement. This mode of operation is called transparent mode and control statements which can direct or specify this mode of operation are called transparent control statements. Transparent control statements are a subset of the total control statements set. The rules for using the transparent control statement are identical with normal control statements with the exception of the additional identification character and the absence of the label option on the transparent control statement.

Summary of Control Statements. - The executive control statement can be divided into eight groups:

Scheduling Statement.

Message Statement.

Input/Output Directive Statement.
Facility or File Handling Statements.
Data Preparation Statements.
Program Execution Statements.
Dynamic Run Stream Modification Statements.
Checkpoint and Restart Statements.

Figure 4 lists all of these control statements in their respective groups and presents a brief description of each statement function. In addition, the following transparent control statements perform the same function as the associated control statements described in the table. @@START, @@LOG, @@ MESSAGE, @@HDG @@SYM, @@BRKPT, @@ASG, @@MSG, @@HDG, @@SYM, @@BLKPT, @@ASG, @@MODE, @@CAT, @@FREE, @@USE, @@QUAL, @@CKPAR, @@RSPAR.

Certain control statements, because of the complexity of their close association with concepts discussed in the Univac supplied manuals, are not discussed in this report. The user is referred to the appropriate documentation for detailed usage information (see figure 2).

File Utility Routines

In addition to the executive control statements, there is a set of control statements recognized by the executive as calls for the File Utility Routine (FURPUR). When the executive encounters a FURPUR control statement, it loads the FURPUR processor. FURPUR continues to process control statements until signaled by the executive that the next control statement is not a FURPUR control statement. Figure 5 summarizes the name and function of each FURPUR statement. The operand field may contain file name, element name or a parameter value depending on the control statement and its use. The FURPUR processor automatically assigns any catalogued file that is not assigned when the FURPUR control statement was encountered. If the FURPUR operation is a modification of the file, the processor places the file in "executive use" state as necessary to carry out the specified operation. After use, the FURPUR operation automatically returns all files to the assigned status the field had before the FURPUR processor control statement was initiated. Temporary files must be assigned by the user. The file utility routines are critical to the operation of the EDIN system because the EDIN system uses the mass storage media of the 1110 series computer for all portions of the EDIN data base.

STATEMENT GROUP	CONTROL STATEMENT	DESCRIPTION
Scheduling Statements	@RUN	Appears at the beginning of each run. Provides accounting, scheduling and run identification information.
	@FIN	Appears at the end of each run.
	@START	Used to initiate the execution of an independent run.
Message Statements	@LOG	Places user specified information in the system log.
	@MSG	Places a message on an operator's (central site) console.
Output Directive Statements	@HDG	Used to place a heading line on print output.
	@SYM	Used to direct nonstandard output action.
	@BRKPT	Used to segment primary output files and to close alternate print files.
Facility Statements	@ASG	Used to assign files (peripheral devices) and catalogued files to a run.
	@CAT	Catalogues mass storage files.
	@FREE	Used to deassign a file and its input/output device or mass storage area.
	@USE	Used to set up a correspondence between internal and external file names.
	@QUAL	Used to define a file name qualifier.

FIGURE 4A SUMMARY OF EXECUTIVE CONTROL STATEMENTS.
(PART 1 OF 2)

STATEMENT GROUP	CONTROL STATEMENT	DESCRIPTION
Dynamic Run Stream Modification Statements	@ADD	Used to dynamically expand the run stream.
	@SETC	Places a value in the condition word.
	@JUMP	Used to bypass a portion of a run stream.
	@TEST	Used to test the condition word when determining portions of the run stream to be processed or bypassed.
Checkpoint and Restart Statements	@CKPT	Used to establish a checkpoint dump that may be used for restart at some future time.
	@CKPAR	Used to establish a program checkpoint dump that may be used for restart at some future time.
	@RSTRT	Used to restart a run at some previously taken checkpoint.
	@RSPAR	Used to restart a program at some previously taken checkpoint.
Program Execution Statements	@NAME	Used to execute a processor.
	@MAP	Used to call the collector and prepare an absolute element.
	@XQT	Used to initiate the execution of a program.
	@EOF	Used to separate data within the run stream.
	@PMD	Used to take edited postmortem and dynamic dumps of the program just executed.
Data Preparation Statements	@ELT	Inserts or updates a program file element from the run stream.
	@DATA	Used to introduce or update a data file from the run stream.
	@END	Used to terminate a data file.
	@FILE	Used to cause the direct creation of a file containing data taken from the run stream.
	@ENDF	Used to terminate the data that follows the @FILE statement.

FIGURE 4B SUMMARY OF EXECUTIVE CONTROL STATEMENTS.
(PART 2 OF 2)

FURPUR CONTROL STATEMENTS	DESCRIPTION
@CHG	Changes element name, version name, file name, read key, write key and mode of a file.
@CLOSE	Writes two hardware EOF marks on a magnetic tape file and rewinds the tape.
@COPIN	Copies elements from an element file located on magnetic tape into a program file on mass storage.
@COPOUT	Copies a program file, or selected elements from a program file, located on mass storage onto a magnetic tape file in element file format.
@COPY	Copies a file or element from one file to another.
@DELETE	Drops catalogued files or marks elements in a program file as deleted.
@ENABLE	Clears the disable flags for catalogued files.
@ERS	Returns to the system all mass storage space allocated to a file.
@FIND	Locates an element in a magnetic tape file (file must be in element file format) and positions the file before the element's label block.
@MARK	Writes two hardware EOF marks on a magnetic tape file and positions the tape between the EOF marks.
@MOVE	Moves a magnetic tape file forward or backward over a specified number of EOF marks.
@PACK	Rewrites an entire program file, removing specified types of elements (depending on the options specified) and all elements marked as deleted.
@PCH	Punches program file elements into 80-column cards.
@PREP	Prepares a program file to be searched for sub-routine references.
@PRT	Provides a listing of the master file directory items for catalogued files, information regarding temporary files, the table of contents of a program file or the text of symbolic elements. Listings of absolute or relocatable elements may be obtained using the LIST processor.
@REWIND	Rewinds magnetic tape files back to the load point of the first real.

FIGURE 5 FURPUR CONTROL STATEMENTS.

Systems Symbolic Processors

Files and elements which contain symbolic data can be processed in various ways using the systems symbolic processors that are available to the user. Available processors are:

1. CULL - which generates a listing of symbols cross-referenced to the element and line in which they are found.
2. DATA - which inserts, updates and corrects data files from within a run stream.
3. ED - which permits conversational editing of symbolic files and elements.
4. ELT - which inserts and updates symbolic elements in a program file from within a run stream.
5. LIST - which generates an edited listing of any type of element.

In addition to the symbolic processors, the @END control statement which operates in conjunction with the DATA and ELT processors, is available.

ELT Processor. - This processor introduces an element into a particular program file and makes corrections to the symbolic element in a program file specified in the run stream. If a new element is specified, the input images to the ELT processor are placed in the run stream following the ELT control statement. If the ELT processor is used to modify an existing file, then modifications placed in the run stream are made to the element. Options are available for processing both data and control statement elements. If control statements are to be processed by the ELT processor, then the D option must be evoked, and the @END control statement must be used after the last statement to be processed. The rules for modifying the existing elements using the ELT (and other data) processors are described in the appropriate Univac documentation (see figure 2).

Data Processor. - The data processor introduces, updates and corrects system data format (SDF) files from the control string. The data processors operations are terminated by the @END control statement who sentential matches are sentential on the data control statement. Any control statement with the exception of the @FIN and @ADD,D control statement may be processed by the data processor. The data processor allows the user to build data files which are an entire or partial run stream. These

files can then be called on by the @START control statement to start an independent run or by the @ADD control statement for inclusion in the current or subsequent runs.

ED Processor. - The ED processor is a TEXT editor which allows the user to conversationally edit the symbolic file or element. It allows insertion, deletion and replacement of text and can merge and split existing files and elements. The ED processor is by far the most versatile and widely used symbolic processing capability on the Univac computer. The ED processor operates in two modes, input and edit. In the input mode all lines entered are directly inserted into the text. In the edit mode, various commands may be used to modify existing text. Changing between modes is accomplished by entering a blank line carriage return. Most editing commands explicitly reference a single part of the text. This is accomplished by an internal cursor maintained by the ED processor. The cursor may be positioned directly by some commands and indirectly by others. A thorough understanding of the use of the ED processor and the associated commands is recommended to the ED user. However, figure 6 provides a brief list of the commonly used edit commands which are available for the construction and maintenance of symbolic files.

CULL Processor. - The CULL processor scans a collection of symbolic elements and produces a cross reference listing of the symbols found; the elements and the lines on which they occur. A list of symbols which are either to be omitted from the sort or the only symbols to be included in the sort may be specified.

LIST Processor. - The LIST processor produces an edited listing of any type of element. For symbolic elements, the image, the line number and the appropriate procedures name table is printed. For relocatable and absolute elements, each text word is printed in twelve digit octal code.

COMMAND	DESCRIPTION
ADD	Adds all or portions of a named file or element to the current file or element. The added information may be in place or at the end of the current file.
CHANGE	Replaces a specified character string with another character string. The action may be specified global or first occurrence only. The action may also be specified for a set number of lines or for all lines beyond the cursor.
DELETE	Deletes lines from the text. The action may be for the current line, a set number of lines or a specified range of line numbers.
DITTO	Duplicates a line or range of lines from within the current file at the cursor position.
INSERT	Inserts a specified string of characters at the cursor position.
LAST	Positions the cursor at the last line of text.
LNP	Prints a specified line or range of lines of text together with line numbers.
LOCATE	Searches the text from the cursor position for a specified string of information.
MOVE	Moves a specified line or range of lines to the cursor position.
NUMBER	(Numeric input) Locates and positions the cursor at the specified line number.

FIGURE 6 COMMON ED PROCESSOR COMMANDS.

The Collector

In addition to the language processor which generates program elements in relocatable form, the Univac 1100 series operating system provides the ability to combine the relocatable elements generated by a language processor into an executable (absolute) element. This combination or collection of relocatable elements is done by the @MAP system processor (or collector). The absolute element produced by the collector is structured such that the executive program loader can place it in execution. Once the absolute program has been created (that is, collected), it may be saved and re-executed many times. The program need only be recollected when a modification to one or more of the relocatable elements is desired.

An absolute element (program) is placed in execution through the use of a program execution control statement (@XQT) within the control stream. When an @XQT control statement is encountered by the executive, the program named in the operand field is retrieved from its mass storage file, loaded into main storage and executed. For a special class of programs (processors), the processor control statement initiates execution.

During execution a program can control which parts of the absolute elements are in main storage by requesting the executive to load previously defined program overlay segments or by linking to program banks. In addition, the program has the capability of attaching to or linking to other previously defined absolute elements. The program has the ability during execution to dynamically control the execution of other semi-independent absolute elements.

The collector is called by the @MAP processor control statement. It provides a direct means of collecting and interconnecting relocatable elements to produce a program in an executable form. This form is called an absolute element. Optionally, the collector can be used to create a single relocatable element from a collection of several relocatable elements. The three basic inputs to the collector are:

1. The parameters supplied on the @MAP control statement.
2. The information supplied by the collector directives.
3. Relocatable elements taken from various sources, such as:
 - a. The temporary program file (TPF\$).

- b. User-created program files.
- c. The system's relocatable library (SYS\$*RLIB\$).

The three basic outputs of the collector are:

1. An absolute or relocatable element.
2. A symbolic element.
3. A program listing.

The primary output of the collector is the relocatable or absolute element which results from the collecting and linking of the various relocatable elements. This element is given a name and placed within a program file for subsequent use. Both the element name and the file in which the element is placed may be dictated by the user.

DATA MANAGEMENT SYSTEM

The EDIN system provides a balance of data management techniques which consider the inherent capabilities of the computer operating system, past efforts in the storage and retrieval of stratified data and the recent development of some flexible paging techniques for the transfer of information between the computer core and the mass storage of the computer. The Univac Exec 8 system provides the resources for the storage of large complex data files, for the storage and retrieval of the files and for the cataloguing protection and backup of the files. The executive system has several processors with instruction sets for manipulating the data retained in mass storage. A limitation on the operating system capabilities arises in accessing the subfile level of information in the system files once the file is addressed.

The EDIN data management system is designed to subdivide the files in a manner that will allow the data which is retained in mass storage to be accessed at any level from the single parameter level to a large matrix of data. Rather than constructing an extensive single computer program that attempts to be everything to everyone, the EDIN data management system provides a three-level data management capability. This approach permits the individual designer using the system to make his own decisions with regard to the storage method and techniques. It also permits the flexibility of using existing data sources not specifically created for EDIN.

The three levels of the EDIN data management system are built upon one another as illustrated in figure 7. The lowest level deals with the interface between the data in mass storage and the computer operating system. The file level of the data management system is provided by the Exec 8 software and consists of the file utility processor FURPUR, the file administration processor SECURE and other system level processors. The system processors are accessed using Exec 8 control statements. Therefore, file level software may be used directly by the designer for transmitting large structured blocks of data or the files themselves to be accessed by the programmer who seeks economy above all else. The file level constitutes the foundation for all higher level data management components.

The second level of the EDIN data management system provides the mechanism whereby the files can be organized into blocks of data called pages. Pages of information can be organized in a number of ways and names can be given to each page. A pointer system or directory is maintained by a Fortran callable software

ORIGINAL PAGE IS
OF POOR QUALITY

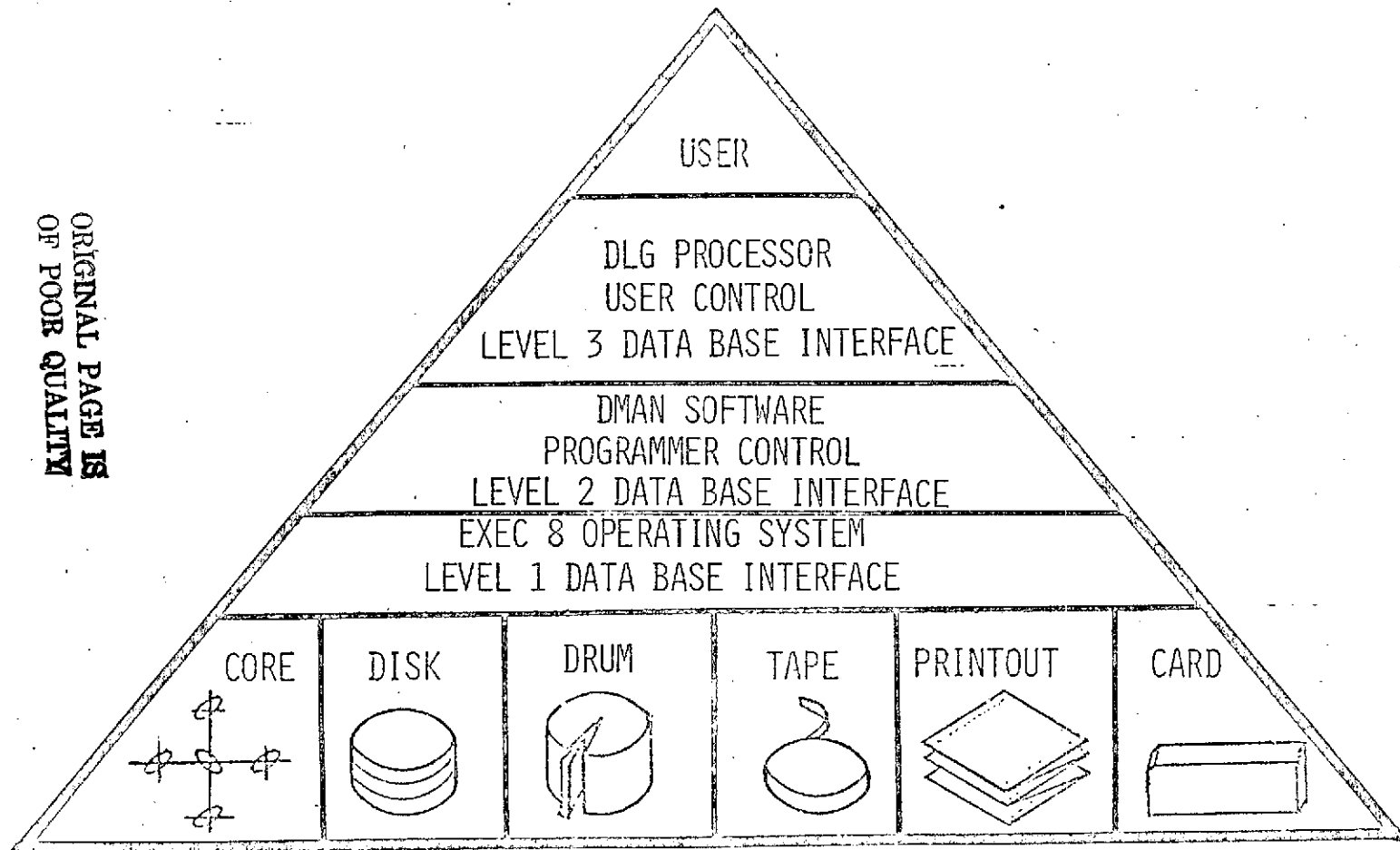


FIGURE 7 EDIN DATA MANAGEMENT SYSTEM.

package, called DMAN, a subroutine utility package maintained in the EDIN library.

The third and highest level of the data management system is provided to make the system more usable to the designer who may not be a programmer. The capability is provided in the DLG processor which is designed to maintain a data base of stratified information, the stratified data can be selectively accessed and merged with the input stream of the EDIN technology programs. This level also provides the interactive language structure which allows the designer to sit at a remote terminal and interact with the data base directly as he develops a design. The DLG processor also contains routines for processing the output from the technology programs for the storage of design information in the data base.

Although the user may access the data base through any of the three levels, it is the lowest level maintained by the Exec 8 system which actually stores and retrieves the data. Exec 8 handles all of the underlying data management functions including file assignments, file directories and maintenance and security procedures as well as the data block transfer to and from mass storage. The Exec 8 system is discussed in the previous section and a thorough treatment of the first level data management is provided by Univac in the appropriate User Documentation (see figure 2). The following discussion deals with the second and third level of the EDIN data management system.

DMAN Software Package

The storage and retrieval of the multitude of data pages which constitute a design data base are managed by DMAN. When a data page is stored, it is given a page name. DMAN keeps a directory of all the names of data pages on a file and the disk addresses where those pages may be found on the file. This makes it possible for a symbolic name rather than a numerical index to be used to access a data page during its residence on the file.

DMAN provides all of the basic data management functions to handle variable length data pages while allowing them to be referenced by name. A data page may be stored on any file which has been established for data base use. All or portions of a data page contents may be retrieved. Modification of the contents of a data page is permitted, including that which requires increasing or decreasing the size of a page. Finally, removal of a data page from a file may be accomplished.

DMAN Usage. - The DMAN data management system is a Fortran callable software package which has been written for access and retrieval of data from the EDIN data base. The package consists of the following subroutines which must be included in the calling program:

DMAN	Basic Read/Write Controller.
NXTAD	Extend File Routine.
UPACK7	Character Unpack Routine.
RITBF	Write Routine.
PACK7	Character Packing Routine.
REDBF	Read Routine.
NWBLK	Create a New Block for Data.

The use requires the following declarations in the user program:

```
COMMON/UNITS/IAREA(273)
DATA IAREA/0,n,271*0/
INTEGER IT(5),IBUF(256)
```

where n is the file number where the data base is stored. The usage is as follows:

```
CALL DMAN(IOP,IT,N,IDATA,IBUF,IAREA(1),IAREA(2))
```

IOP	The read/write option. A further discussion of these options is given later.
IT	A five word array containing the data title. A further discussion of the titles is given below.
N	This variable contains the number of words in IDATA to be read or written. When reading, and the requested list cannot be satisfied, this value is reset to the number of words actually read, so this item must always be a variable when reading data.
IDATA	An integer or real array containing the data to be stored in the data base. There is no restriction on the length of this array.
IBUF	A 256 word buffer area for use by DMAN.

IAREA This is a unit dependent area needed by DMAN. It must be dimensioned 273. One IAREA is required for each unit using DMAN. The double appearance of this array in the calling sequence is required for internal addressing purposes. This area must be protected, such as in COMMON, and must be reserved for use by DMAN while this file is being used.

A Discussion of IT. - There are two significant portions to the five word array IT. The first three words of the title are user supplied hollerith words which represent the name of the data item which is to be accessed or stored in the data base. If this is the first access of this data in the data base, the fourth word must be set to zero. This zeroing of the fourth title word will also return access to the beginning of the data set stored under the title given in the first three words.

The fourth and fifth words of the title are reserved for use by DMAN. If the fourth word is zero, a search is made of index arrays to find the address of the desired data set. This address is then inserted into these two words. Each time some activity occurs using this title, the address stored in these two words is updated so that this address always refers to the next word after the last word accessed. This eliminates the need to search the index arrays for each access of the data.

A Discussion of IOP. - IOP controls the type of reading or writing done by DMAN. The I/O options are:

IOP = 10 - write a matrix. The complete data set to be stored under the title IT is present in IDATA.
 = -10 - read a matrix.
 = 20 - write a single fixed length record.
 = -20 - read a single fixed length record.

 = 21 - write a single variable length record. Using this type of write option, an end-of-record mark is inserted after the end of the record. Any variable length record read will not pass this mark when reading. If the read is a fixed length record read, however, this mark will be ignored.

= -21 - read a variable length record. In this case, N is the number of words requested. The read will continue until N words have been read, and end-of-record mark is found, or the data set is exhausted, whichever comes first. The value of N will be set to the number of words actually returned.

= 30 - extend a data set with a fixed length record. The data in IDATA is to be appended to the existing data set stored under the title in IT.

= 31 - extend a data set with a variable length record.

NOTE: If a read attempt is made, which will extend the read past the end of the stored data set, or the data set requested has not been stored, the following values will be returned by DMAN:

N=0 and IDATA(1)=3LEOD.

IOP = 6HPURGE - this option will cause the title given in IT to be purged from the index array.

IOP = 6HCLEAR - this action will cause the buffer IBUF to be cleared. That is output to disc if necessary. This action is necessary before releasing the buffer to other uses, or existing a subroutine or overlay under conditions which will not protect the buffer.

IOP = 6HCLOSE - this action conditions the data base so that the entire contents of the data base do in fact reside on disc. It is necessary to execute this statement on any catalogued data base to insure that its entire contents are on disc. Normal activity may proceed after the function is called, and this function may be called as many times as desired.

The DLG Processor

The level three (3) data management is provided by a user controlled processor called DLG. The DLG processor is an 1100 series Exec 8 computer program designed to read, modify, manipulate and replace symbolic images. DLG is controlled by a set of user supplied directives (or language elements) which provide the basic capabilities of the DLG processor as follows:

1. Language elements for the construction and maintenance of a data base which is independent of any other computer program.
2. Language elements for processing information generated by other computer programs.
3. Language elements for automatically retrieving data base information as input to any other computer program.
4. A simple technique for editing and interrogating the data base and for generating summary reports of data base information.

The DLG processor can be used to form a linkage between engineering technology modules through the manipulation of common information in the data base. The use of the system for this purpose requires the prior assimilation of the following basic components:

1. A library of independent technology programs including the DLG processor.
2. The control card sequences for the execution of the technology modules.
3. The setup data for the technology modules which perform the desired analysis function.

The basic components are often available without additional program development. The program sequencing and intercommunication required to integrate the basic components into a design simulation are established using the Exec 8 run stream concept. Data base requests for common information are established by the formation of skeletonized data elements containing execution control cards and/or technology module input data. The skeleton elements are processed or filled out by the DLG processor using data base information.

The procedure for implementing the linkage is to read output data from one module, insert a selected subset of the resulting data into a stratified data base and then selectively extract this and other stored data for placement into the input stream for other applications programs. The linkage is illustrated in figure 8. The effect is to provide a unified analysis involving several modules operating from a single source of data. Repetition of execution sequences can be triggered by looping criteria residing in the data base.

DLG Usage

Control Statement.-

```
@DLG.DLG,options lfn.eltl,lfn.elt2
```

Option Specifications. -

- I Source input will follow the processor card.
Source output will be placed in eltl.
- L Source input data will be listed.
- O Source output data will be listed.
- D Card cracking information will be listed.
- E Solicitation and result of directives will be printed.
- S List interrupt mode will be invoked.
- M New data base files will be generated with this execution.
- B Build option will be invoked. This option specifies that all data directives of the form:

```
'name name=value---
```

or

```
$name name=value---
```

This will permit the addition of data to the data base regardless of the directive name. Otherwise, only those data base variable names, which were previously defined in the data base, will be updated unless the data directive name is ADD or DEFINE.

The B option may not be invoked via the "ON" command. If desired, it must be present on the processor call card.

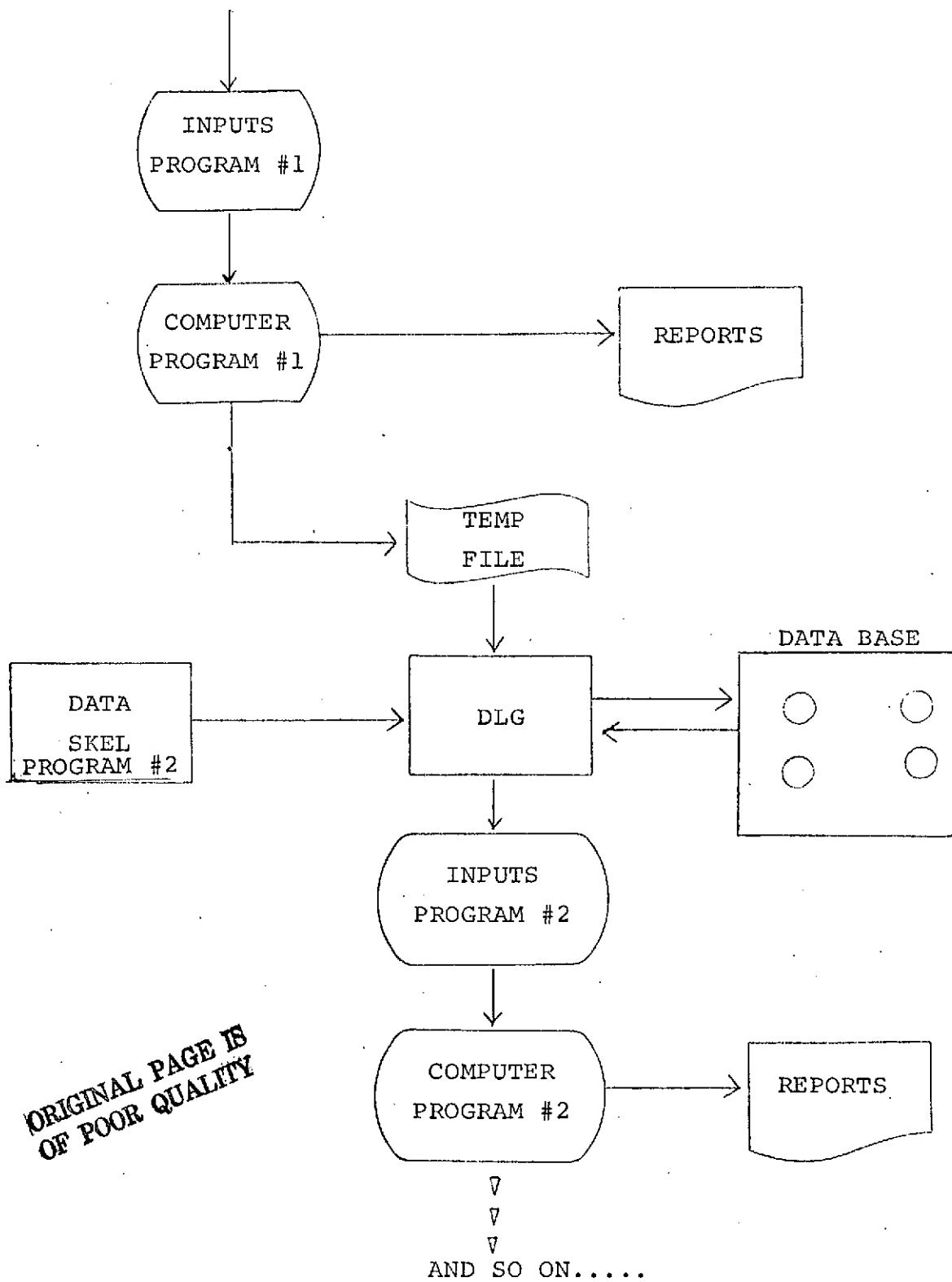


FIGURE 8

USE OF THE DLG PROCESSOR.

Syntax Definition. -

name	Must be six (6) or less alphanumeric characters and begin with an alphabetical character.
'(quote or prime)	The DLG delimiter. Strings that occur between pairs of delimiters will be processed by DLG. Strings external to primes will be passed "as is" into the output element.
-	The underline on a command indicates an optional character string which may be used as a directive.
value	Indicates a data base value in real, integer or hollerith format.
i,j,k	Indicates integer constants used in the directives.
elt	Exec 8 file element name in program file format.
lfn	Exec 8 logical file name in system data format.
text	Textual information.
[]	Indicates optional items on the line.

Summary of DLG Directives. - The DLG directives are summarized below. Underlines are optional character strings. All commands are excluded data base names.

'name'	Replace name with information from the data base.
'ADD	Replace specified information in the data base.
' <u>CHANGE</u>	Change values in common IDLOG.
' <u>COMMENT</u> or '.	User description with null effect.
' <u>CREATE</u>	Create a new data base.
'CSF or 'ER	Submit executive control statement.
' <u>DBLIST</u>	Print the names of all random access data bases on the data base file.

'DEFINE Place description in data base directory.
'FORMAT Format free data base information in place.
'INSERT Insert binary SDF data in place.
'ON Mode activation.
'OFF Mode suppression.
'PRINT Print data base information.
'USE Specify a circular data base search.
'UPDATE Update a specified data base.

Descriptions of Control Directives. -

'ADD name' - Specifies that information will be added to data base.

'ADD name=value'
 'ADD name=name'
 'ADD name=value,value,---'
 'ADD name=name,name,---'
 'ADD name=name op name, name op value,---'

	+	Add
	-	Subtract
where op =	/	Divide
	*	Multiply
	**	Exponentiation

'CHANGE number=value' - Using the integer number 'number' as an index into the master common block, IDILOG, the current value is replaced by 'value.'

'COMMENT _____' - This is a null card and is discarded by DLG.

'CREATE name,DIRLEN=number,LENDES=number,LTOTAL=number' - The data of name 'name' is brought into existence on the data base file. Optional parameters are DIRLEN - the directory length (This should be a prime number.).

LENDES - Length in computer words of the description.

LTOTAL - Total size, in computer words, reserved for the data base.

'CHANGE' -

Example 'CHANGE 27=3'

Location 27 of the common block IDILOG will have its value replaced by an integer 3.

'COMMENT' - A null card. The delimited field is removed from the card. If the resulting card is BLANK, the card will be removed from the run stream.

'CSF @ Control Statement' - Specifies that an execution control statement will be processed using the standard CSF\$ package. The following control statements may be used:

@ADD	@CKPT	@RSPAR
@ASG	@FREE	@RSTRT
@BRKPT	@LOG	@START
@CAT	@MODE	@SYM
@CKPAR	@QUAL	@USE

Example:

```
'CSF @USE 25, DBASE'
'CSF @ADD DUSEFIL.DLOG'
'CSF @QUAL B'
```

'DEFINE name=value,text' - Stores a textual description with the name in the data base directory. If the name is a new directory entry, the value is the number of data base entries allotted. Existing data is unaffected and new data is not added.

'DEFINE A, LETTER 1' - Stores the description, LETTER 1, with the name A.

'DEFINE B=10, BARRAY' - Stores the description, BARRAY, with the variable name B and allots 10 data base entries for B.

'FORMAT name=value/value, (Fortran compatible format statement)' Extracts freely stored data from the data base and places into the output elements in accordance with the given format.

```
'FORMAT A=6/3, (1X, 3F15.3)'
```

The six items of A are output into the named element, 3 on a line through the (1X,3F15.3) format.

'INSERT name=value/value' - Specifies that binary coded information the SDF file name will be placed in the source output element in 14A6 format.

'INSERT A' - Entire file of data in A will be transferred to source output element.

'INSERT B=5-13' - Insert data from B from records 5 through 23.

'INSERT C=5*EOF' - Insert records from file C records 5 to the end-of-file.

Other Examples - 'INSERT A,B=5-23, C=5*EOF'

'name' - Specifies a simple replacement of named information with data base parameters or arrays.

'REAL'	Real parameter or array.
'INTEG'	Integer parameter or array.
'HOLITH'	Hollerith parameter or array.
'LOGICL'	Logical parameter or array.
'ARRAY(j)'	Real or integer element of an array, j must be a constant greater than 1. A value of j=1 will cause the transfer of all of j.

'ON name,name---' - Mode activation directive.

'OFF name,name---' - Mode suppression directive.

P or PAGDMP	Print card cracking information.
O or OUTDMP	List logical file 1 data.
N or INDUMP	List source output element.
C or CONTINUE	Activate continuation card option.
L or LIST	List source input information.
S or SPLIT	Interrupt mode.
E or EDIT	Edit mode (demand response to printer).

'PRINT name' - Specifies that data information will be printed.

'PRINT name=A,Z'	Print all information in name.
'PRINT name=n,m'	Print entries n through m alphabetically.
'PRINT name'	Directory and first data base entry of named data base.
'PRINT'	Directory and first data base entry of current 'USE' assigned data bases.

'USE' -

'USE A,B,C' - The data bases named will be circularly searched in the order given for variables used in replacements. All will be searched once before a NO FIND is declared. It should be noted that this command may cause very excessive SUP changes if not carefully used.

'UPDATE name' - Specifies that the named data base will be updated with the information which follows:

'UPDATE A' - Specifies that the data base A will be updated with the data which follows.

Processor Interface

The processor interface is a Univac 1100 series EX8 utility subroutine, IF, written in assembly language. IF is designed for use by the FORTRAN programmer in the construction of a processor. It allows the information on the processor call card to be made available to the user program. Two fields on the processor call card are available to the user. The first is the input field, and the second is the output field. The I option implies only the first field will be used. This field will be the output field.

Usage. - The programmer is assumed to have a minimum working knowledge of Univac's EXEC 8 operating system and the use of such system processors as ELT, FOR and FURPUR. There are three entry points into the subroutine: SIREAD for reading from the SI field, PGMOUT for writing to the SO field, and DONE for closing the file. The calling sequences and the associated arguments are as follows:

CALL SIREAD (\$err,\$eof,IMAGE,'word')

\$err Statement number to be transferred to in case of error.

\$eof Statement number to be transferred to when an end-of-file is reached.

IMAGE An array containing the image you want written out. Normally, this is 14 words long.

'word' This word is used to delete words from the right, back to the left to make the image as short as possible in order to conserve disk space. For card images, this would be a word of blanks: for binary information in internal machine format, zero would be best.

SIREAD Stands for source input read.

CALL PGMOUT (\$err,\$eof,IMAGE,'word')

\$err Statement number of location to be transferred to in the event of an I/O error.

\$eof	A dummy argument.
IMAGE	The array containing the image of words to be written out (normally dimensioned 14).
'word'	As the image is compressed on disk with the trailing null words dropped, this word is used to fill out the image so that when it is returned to the user, it is the full 14 words long.
PGMOUT	Stands for <u>program output</u> .
CALL DONE (\$err)	<p>This call must be executed prior to conclusion of the program. It will drain any uncompleted buffers, close and release to their original status any attached files. If this entry is not called prior to program termination, the created element of SO field will not be properly created.</p>

Restrictions. - There are three important limitations on the use of IF:

1. There can be only 2 fields on the processor card.
2. SI READ must be called prior to any reads from the standard system input device, the card reader (unit 5 in FORTRAN); otherwise, read errors will occur.
3. Once the entry DONE is called, none of the entries into IF may again be referenced (the program will error off if this rule is violated).

Technology Module Interface Package

The communication of information from a technology program to the EDIN data base generally requires modification of the applications program. This modification is usually trivial and requires little programming knowledge to accomplish. The objective of the modification is to create a special file of information which contains a format suitable for reading by the DLG processor. The information is placed on the special file by the technology program. The file is later integrated by the DLG for possible placement of the information into the EDIN data base.

A series of four routines for printing the common types of data in a format readable by DLG are available. They may be called at any point in the calculation sequence for generating EDIN output. The format simulates the control directives format used in the DLG processor.

ADDREL - For printing real variables and arrays.

ADDINT - For printing integer variables and arrays.

ADDHOL - For printing Hollerith variables and arrays.

ADDLOG - For printing Logical variables and arrays.

The output is similar to the format of NAMELIST for one variable name only with any number of associated values. Each subroutine has the same calling sequence characterized as follows:

CALL ADDREL (LU, NAME, NUM, VALUE)

LU - Logical unit or special output file.

NAME - Desired name chosen by the analyst/programmer.
It may be a stored name set by a Fortran data statement or can be set in the calling sequence as nHname.

NUM - Number of values in the array. For a single variable NUM=1.

VALUE - Internal variable or array name (starting location).

The subroutines for the other variable types have the same calling sequence. The primary difference among them is the format used for writing the variables and the special output file. Each output is a DLG control directive format. The name associated with the directive is set by a data statement in the individual subroutines. The data statement may be set at the time the

technology program is modified. Usually it is desirable to use a name which is reminiscent of the application program name. The selected name may be precisely the same as the acronym used to execute the application program in EDIN. The reason for such a choice is that the directive name is stored in the EDIN data base. A print of the data base prints the last directive which updated each variable in the data base.

For most technology programs, the use of the software described above is adequate. However, certain programs generate data base information in a Fortran "DO LOOP." In these instances, the package (by itself) can not satisfy the EDIN requirement of separate names for different data elements and arrays.

The most convenient way to make this program and others of this type compatible with EDIN is to provide some name-generating capability with the applications program. Function subroutines which provide this capability can be called as illustrated below:

NAMGEN (NAME, K, J)

NAME = The desired root name.

I = Concatenated number occupying the first one or two BCD character positions beyond the root name.

J = Concatenated number occupying the second one or two BCD character positions beyond the root name.

An example would be:

NAM=NAMGEN (4HNAME,1,2)

In the above illustration, the name NAME would be extended by the BCD characters 1 and 2 concatenated to it and stored in NAM.

NAM=6HNAME12

A maximum of 6 characters may be generated. This limit is imposed by the word size limit for EDIN data base names.

Usually the NAMGEN function is used in conjunction with the NAMELIST simulator described above in the following manner:

CALL ADDREL(LU,NAMGEN(NAME,I,J),NUM,VALUE)

In the illustration, the name is generated within the calling sequence of the subroutine which prints the simulated namelist for the generated name.

PROGRAM LIBRARY

The following sections describe the current EDIN program library. The independent program elements, which form the library, are resident on the Exec 8 system or are readily available for incorporation should they be required. Many of the programs are nationally recognized, references 17 through 30. Others were specifically developed for the EDIN system.

The library is arranged according to the specific groups of application's software. These groups are defined in the following paragraphs. In addition, a series of abstracts are provided on the software programs available to the EDIN system.

Geometry

The geometry applications software is a collection of specialized geometry generators. Software which generates geometry or uses formatted geometry as an input is included in this classification. The following list of geometry application software and descriptions are available in the EDIN library.

- AIRFOIL: A Program for Generating Geometric and Aerodynamic Characteristics of Airfoil Sections.
- GEOMETRY: Body Coordinate Generator.
- PANEL: A Program for Generating Panelled Configuration Geometry.
- VAMP: Volume, Area and Mass Properties.
- WETED: Wetted Area and Reference Length Program.
- SFIT: A Surface Fitting Program which Generates Surfaces over Cross-Sectional Definitions. The Surfaces Generated are Defined by Cornerpoint Geometry.
- AIRCFT: Program to Generate Aircraft Type Configuration Geometry.
- GTM: Geometry Technology Module for Generation, Manipulation and Display of Cornerpoint Geometry.

Aerodynamics, Stability and Control

The Aerodynamics, Stability and Control applications software are those programs which perform configuration design analysis for predicting stability and control derivatives and aerodynamic

properties through various flight regimes. These programs and program descriptions are summarized as follows:

AIRFOIL: A Program for Generating Geometric and Aerodynamic Characteristics of Airfoil Sections.
DATCOM: Configuration Design Analysis Program (TRW).
DATCOM2: Configuration Design Analysis Program (MDAC).
HABACP: Hypersonic Arbitrary Body Aerodynamic Computer Program.
SKINF: Turbulance Skin Friction Drag Program.
TOLAND: High Lift Aerodynamics Prediction.
TREND: Subsonic/Supersonic/Hypersonic Aerodynamic Tradeoff Program.
WDRAG: Zero-Lift Wave Drag Program.
WETTED: Wetted Area and Reference Length Program.

Propulsion

The propulsion applications software would include those programs which are used for rocket engine sizing, thrust chamber design, turboramjet engine design and inlet design. These programs are summarized as follows:

ENCYCL: Design Point Performance of Turbojet and Turbofan Engine.
LREN: Liquid Rocket Engine Model.

Mass and Volumetric Properties

The Mass and Volumetric Properties applications software include those programs which perform weights trend analysis, volume and mass properties evaluations and sizing. These programs are summarized as follows:

ASPE: Weights Trend Analysis for Multiple Stage Vehicles.
CASPER: Reusable Booster/Orbiter Sizing.
CAWATA: Cost and Weight Analysis of Transport Aircraft.
ESPER: Weights Analysis for Shuttle Class Vehicle.
SSSP: Space Shuttle Synthesis Program.
VAMP: Volume, Area and Mass Properties using Harris Geometry.
VSAC: Vehicle Synthesis for High Speed Aircraft.

WAATS: Weights Analysis for Advanced Transportation Systems.

TANK: A Program for Generating Fuel and Oxidizer Tank Design Characteristics such as Volume, Wall Thickness and Weight. Cornerpoint Geometry is Generated for Display Purposes.

SIZER: A Preliminary Booster Sizing Program Based upon Ideal Velocity and Mass Ratio Relationships.

WAB: A Program that Computes Volume, Area and Mass Properties from any Combination of Gentry Geometry and Black Box Inputs.

Performance

The performance applications software includes programs which perform trajectory optimization and analysis and design synthesis. These programs are summarized as follows:

GEMASS: Three Degree/Six Degree of Freedom Motion Analysis Program.

PADS: Performance Analysis and Design Synthesis Program.

POST: A Program to Optimize Simulated Trajectories.

PRESTO: Program for Rapid Earth-to-Space Trajectory Optimization.

ROBOT: Three Degree of Freedom Launch Optimization Program.

SSFS: Space Shuttle Functional Simulator.

SVDS: Eighteen Degree of Freedom Multiple Vehicle Motion Analysis.

TOLAND: Takeoff and Landing Program.

VSAC: Vehicle Synthesis for High Speed Aircraft.

Thermodynamics

The following application software is available for thermodynamic evaluations and analysis:

HABACP: Hypersonic Arbitrary Body Aerodynamic Computer Program.

TREND: Subsonic/Supersonic/Hypersonic Aerodynamic Tradeoff Program.

Structures

The structures application software includes those programs which perform structural analysis and structural optimization of vehicles. These programs are summarized as follows:

AFSP: Automated Flutter Solution Procedure.
ASOP: Automated Structural Optimization Program.
BOSOR: Buckling of Shells of Revolution.
CALBAR: Buckling Loads/Cylinder Shells.
SSAM: Swept Strip Aeroelastic Model.
STAGS: Structural Analysis of General Shells.

Cost

The following application software is available for performing cost analysis:

CAWATA: Cost and Weight Analysis of Transport Aircraft.
DAPCA: Development and Production Cost of Aircraft.
PRICE: A Program for Improved Cost Estimation.

Environmental Protection

The following application software is available for predicting environmental protection requirements and environmental flight evaluations.

SBOOM: Program to Predict Ground Level Overpressure from the Over-Flight of Shuttle Type Vehicles.

General and Special Purpose Utilities

The following general and special purpose utility software are available:

PLOTTR: A General Purpose Plotter Program which Outputs to Tektronix, CALCOMP, SD4060, MOPS.
CIPHER: A Report Generation Program.
AESOP: Automated Engineering and Scientific Optimization Program.

DLG: Data Processor for Linking Independent Computer Programs and Providing Data Intercommunication.

HDG: Heading Program.

MAC: Macro-Processor.

IMAGE: A Program for Displaying Three Dimensional Geometric Configurations on Tektronix, CALCOMP, SD4060, MOPS, Etc. This Program Can Perform Rotations, Translations and Zooming.

VL70: Program to Interrogate Standard Aero Tape Data.

PROFIL: Program to Interrogate Trajectory Data Tapes.

The following is a collection of abstracts on the applications software available to the EDIN system:

ABLATOR: One Dimensional Analysis of the Transient Response of Thermal Protection Systems.

The program assumes that thermal properties in the given layer of material are functions only of temperature, that all heat flow is normal to the surface, and that gases transpiring to the char layer are the same temperature as the char. The outer surface is subjected to the aerodynamic heating and the char layer provides both installation and high temperature outer surface for reradiation. The heat passing through the layer is partially observed by pyrolysis at the interface between the char layer and the uncharred material. The remaining heat is conducted into the uncharred layer. The gases generated through the char layer are injected into the boundary layer. The gases are heated as they pass through the char and this heat removal from the char layer induces the quantity of heat conducted by the pyrolysis interface. When these gases are injected into the boundary layer, the conducted heat transfers are induced. The program accepts as input the temperature at the surface of the model and generates a time history of the thermal condition of the various layers as the vehicle enters the atmosphere.

ACMOTAN: Linear Aircraft Motion Analysis

The program is a versatile code for linear aircraft motion analysis which allows the user to supplement the standard airplane equations of motion with auxiliary equations written by the user to represent control laws or additional variables. The program prepares the system of linear differential equations using several optional forms of input data and then carries the solution to an extent determined by the output option selected. Minimum output includes the characteristic polynomial and its roots. Additional output in the form of transfer functions, frequently responses and time histories can be selected.

AESOP: Automated Engineering and
Scientific Optimization Program.

AESOP is a multiple variable optimization program designed for the solution of a wide range of parameter optimization problems. The basic program has the ability to solve constrained optimization problems involving up to 100 parameters and up to 20 constraints. Thirteen search techniques are available for use individually or in combination to solve the desired problem. The methods include sectioning, steepest descent, quadrilateral search, Davidon's method, random ray search, pattern and several others. The program is designed to be linked with other programs to perform internal optimization or can be used as an independent program for optimizing systems of programs.

AFSP: Automated Flutter Solution Procedure

Program AFSP is based upon a new method of solving the flutter equation. The method is based upon the premise that the flutter analysis, due to the particular form in which the aerodynamic forces are available, essentially consists of a search for those V-values and w-values which render the flutter determinate zero. The procedure deviates essentially from the V-g analysis in so far that the eigenvalue calculation is replaced by simpler algorithm leading to the decomposition of the flutter matrix. The actual search for the flutter solution involves a single real and positive function of the two variables V and w rather than the (n) complex functions representing the eigenvalues in the V-g analysis. The flight altitude, Mach number and flight speed remain consistent throughout the search whereas the V-g analysis starts out from given values at altitude and Mach number. The flutter speed only follows as a result of the calculation. In general, the speed will not be consistent with the input data such that many runs are required to iterate toward a solution.

AIRFOIL: A Program for Generating Geometric and Aerodynamic Characteristics of Airfoil Sections.

The computer program is written to provide airfoil coordinates incompressible inviscid section characteristics and two-dimensional drag-rise Mach numbers for a large number of National Advisory Committee for Aeronautics (NACA) designated airfoils from a simple one card input. The program is actually a combination of two separate programs. One program gives the airfoil surface coordinates with only the NACA airfoil designation as input, and the other uses the surface coordinates to predict incompressible, inviscid pressure distribution from which the section characteristics and drag-rise Mach number are determined.

ATOPII: Atmospheric Trajectory Optimization

The program is a generalized steepest descent computer program set up to handle the three-dimensional, point mass, vehicle flight path trajectory optimization problem. It is capable of simultaneously handling up to fifteen state variables, six control variables and ten constraints. Most of the usual functions required in flight path studies are available within the program; others may be added as desired by simple program additions, providing the function or its derivative is defined analytically. The program may be readily extended to cover steepest descent optimization problems in other fields, by the replacement of the basic differential equation subroutine by any other set of equations of the same general type. Convergence to the optimal solution is obtained automatically by means of one of two control systems which, by a series of logical decisions, obtain a reasonable perturbation magnitude at each iteration.

COAP: Combat Optimization and Analysis Program.

The program is an extension of the ATOP II (Atmospheric Trajectory Optimization Program). It uses two complete three dimensional equations of motion sets to simulate a one-on-one combative encounter between two flight vehicles. The aerodynamic and propulsion representation are sufficiently general to permit the simulation of both current and proposed vehicles by input data. Generalized rotating planetary and atmospheric models permit stimulation of either aircraft, missiles or spacecraft encounters. Combat roles for each vehicle (attacker, defender, etc.) are automatically defined on the basis of vehicle relative positions, heading and velocities. Depending upon the vehicle role selected, any one of the set of tactics designed to satisfy the role requirement is executed. The tactics vary in nature from straight forward stylized maneuvers such as split-S to barrel role, to three-dimensional lag or lead pursuit path. Combat optimization capability may be introduced by repetitive simulation using parameterization of the combat guidance parameters and the application of multivariable search techniques. Alternately, the variational calculus may be employed to define optimal continuous control against a reacting opponent. In the parameter optimization mode, the option to determine a mini-max solution is also available.

CONPLOT: Aircraft Configuration Plot.

This program generates the necessary instructions for automatically plotting of a numerical model of an aircraft configuration. Program options may be used to draw three-view and oblique orthographic projections as well as perspective drawings of an aircraft. These plots are useful in checking the accuracy of the numerical model data. Magnetic tape output from this program has been used

DAPCA: Development and Production Cost of Aircraft.

The DAPCA program computes development and production costs of major subsystem of fly away aircraft (airframe, engines, etc.). The cost input data is simple and generally relates to aircraft and engine performance characteristics such as gross takeoff weight, speed engine type thrust, etc. The actual cost equations are the power law types with built-in cost coefficient and using user supplied parameters.

DATCOM: Configuration Design Analysis Program (TRW).

The program computes aerodynamic coefficients for aircraft/spacecraft configurations in the subsonic/transonic/supersonic regimes. Analytical techniques in the program are based on those of USAF stability and control handbook, DATCOM, revision September, 1970. The program comprises four modules which compute lift, pitch, side-slip and control characteristics respectively. Modular construction enables other sets of aerodynamic characteristics to be incorporated into the program.

DATCOM2: Configuration Design Analysis Program (MDAC).

The program calculates the statics stability characteristics of wings, bodies, wing-body, tail-body and wing-body-tail combinations at angle of attack and side slip through the Mach range from subsonic to supersonic speeds. Whenever appropriate DATCOM methods are available, the program computes longitudinal derivatives C_{L_α} and C_{M_α} , longitudinal coefficient C_D , C_L , C_m and side slip derivatives C_{Y_β} , C_{l_β} and C_{n_β} . Output for configurations of horizontal tails also include downwash and dynamic pressure values. All intermediate variable calculations are also available for output.

ENCYCL: Design Point Performance of Turbo-
jet and Turbofan Engine.

ENCYCL computes the design point performance of turbojet and turbofan engine cycles from user supplied engine characteristics and flight conditions. The program input requires the airplane Mach number, the altitude, the state conditions, turbine inlet temperature, afterburner temperature, duct burner temperature, bypass ratio, coolant flow, component efficiencies and component pressure ratios. The output yields specific thrust and specific fuel consumption, engine efficiencies and several component temperatures and pressures. The thermodynamic properties of the gas are expressed as functions of the temperature and fuel to air ratio.

GENENG: A Program for Calculating Design
and Off-Design Performance for
Turbojet and Turbofan Engines.

The program calculates steady state design and off-design performance for one and two spool turbojet engines. The original version of the GENENG program entitled, "Simulation of Turbofan Engines" was developed by the Turboengine Division of the Air Force Aero Propulsion Laboratory, Wright Patterson Air Force Base, Ohio. The program uses steady state gas dynamics to compute the engine design conditions. Off-design performance is based on specific component performance maps which must be provided by the user.

GENENGII: A Program for Calculating Design and
Off-Design Performance of Two and Three
Spool Turbofans with as many as Three
Nozzles.

The GENENGII program is a derivative of GENENG (Generalized Engine Program). GENENG is capable of calculating steady state design and off-design performance of turbofan and turbojet engines were evolved from SMOTE (Simulation of Turbofan Engine) which was developed by the Turbo Engine Division of the Air Force Aero Propulsion Laboratory of Wright Patterson Air Force Base, Ohio. GENENGII calculates design and off-design engine performance for existing or theoretical fan engines with two or three spools and with one, two or three nozzles. In addition, fan performance can also be calculated. Nine basic turbofan engines can be calculated without any programming changes. Included among the nine are three types which are likely candidates for STOL aircraft with internally blown flaps. Many other possibilities exist which are too numerous to mention, being determined by the users knowledge of the program itself.

GEOMETRY: Body Coordinate Generator.

The program generates trapezoidal and elliptical body coordinates in a format suitable for use in VAMP, WETTED, DRAG and CONPLOT. The forbody coordinates are generated according to a "minimum drag" area distribution while the afterbody coordinates are constant in cross sectional area.

HABACP: Hypersonic Arbitrary Body Aerodynamic Computer Program.

The program treats the vehicle surface as a collection of quadrilateral elements oriented tangential to the local vehicle surface. Each individual panel may have its local pressure coefficient specified by any of a variety of pressure calculation methods, including modified Newtonian, blunt body, Newtonian-Prandtl-Meyer, tangent-wedge, tangent-cone, boundary layer induced pressures, free molecular flow and a number of empirical relationships. Viscous forces may also be calculated, which include viscous-inviscid interaction effects. Skin friction options include the reference temperature and referenced enthalpy methods for both laminar and turbulent flow, the Spalding-Chi method and a special blunt body skin friction method. Control surface deflection pressures including separation effects that may be caused by the deflected surface are also calculated. Several other options are available including the calculation of dynamic derivatives, the generation of geometry and plotting.

HDG: Heading Program.

HEADING is a control card callable program which prints user specified heading information in large characters. The characters of the heading are formed by the pattern of characters which form the character itself. The letters of the heading are eight characters wide and ten characters high. Input to the program is placed on the heading execution card.

IMAGE: Configuration Display Program.

The IMAGE program uses a surface definition based on quadrilateral elements to describe picture-like drawings of arbitrary configurations. The program is used for visual check on geometric input data, monitoring of geometric perturbations and providing reports on geometric characteristics. Geometric characteristics may be input or taken from a data base of configuration data. The user describes the viewing angles, position and scaling factors as well as textual information through the input procedure. The configuration drawings are generated on a plot vector file which is suitable for processing by various display devices.

MINIVER: Aerodynamic Heating Program.

The program is a "miniature version" of the McDonnell Douglas Corporation Aerodynamic Heating Program for use on the CDC 6600. It calculates radiation equilibrium temperature and provides thin-skin temperature response for input materials. (Aluminium, titanium, Rene 41 and inconel X-750 properties are built in.) Heat transfer methods are available including Fay and Rydell, Erkart Reference Enthalpy, Spaulding and Chi, Flat Plate Rho-Mu product, three swept cylinder theories and Lees-Detra and Hidalgo for hemispheric nose caps. The program also includes the capability of traversing three sequential shocks, shape-edge and cone plus oblique shock of availability, sharp-edge and cone modified Newtonian and swept cylinder stagnation line pressure solution.

PADS: Performance Analysis and
Design Synthesis Program.

The Performance Analysis and Design Synthesis (PADS) computer program has a two-fold purpose. It can size launch vehicles in conjunction with calculus of variations optimal trajectories and can also be used as a general purpose branched trajectory optimization program. In the former case, it has the Space Shuttle Synthesis Program as well as a simplified stage weight module for optimally sizing manned recoverable launch vehicles. For trajectory optimization alone or with sizing, PADS has two trajectory modules. The first trajectory module uses the method of steepest descent, the second employs the method of quasilinearization, which requires a starting solution from the first trajectory module.

PANEL: A Program for Generating Panelled
Configuration Geometry.

The PANEL program is a general purpose external geometry definition program developed primarily for use in large scale simulations of the preliminary design process. It is an independently operated program which produces a sequence of quadrilateral panels defined by the corner points. The resulting data is acceptable as input to computer programs in other technical disciplines such as aerodynamics, structures and thermodynamics.

The program accepts as input a variety of formats from detailed definition of individual panels to selection of generalized two- and-three dimensional shapes. The section data includes circular and elliptical as well as arbitrary cross sections. A cubic patch technique is included which allows broad sections of the vehicle to be described with a relatively small amount of input. The input data can be mathematically fitted with end matched cubic functions and reduced to distributed panels.

PLTVIEW: Program for Generating
Separation Plots.

The program is a companion to the SEPARTE program which computes the separation distance and attitude of two stages of the shuttle vehicle during staging. PLTVIEW generates plots of separation distance and vehicle orientation for the separating vehicle components. The input data is obtained from the time history data generated by the SEPARTE program. The output is a GERBER plot of the separation distance and orientation.

POST: A Program to Optimize
Simulated Trajectories.

The POST program is a generalized point mass discrete parameter targeting an optimization program. POST provides the capability to target and optimize point mass trajectories for a powered or unpowered vehicle near an arbitrary rotating oblate planet. POST has been successfully used in solving a wide variety of atmospheric ascent and re-entry problems as well as exoatmospheric orbital transfer problems. The generality of the program is evidenced by its N-phase simulation capability which features generalized planet and vehicle models. This flexible simulation capability is augmented by an efficient, discrete parameter optimization capability which includes equality and inequality constraints.

RDPRO: POST Plot Data
Generation Program.

The program interrogates the output data tape from the POST program and generates specified plot information pertaining to time histories of various performance and constraint functions available after the execution of the POST program. The program is designed specifically as an interface program for analysis of trajectory data from the POST program.

PRESTO: Program for Rapid Earth-to-Space Trajectory Optimization.

The PRESTO program uses a closed loop deepest descent optimization procedure to derive flight trajectories to produce maximum booster payloads for a variety of space missions. Trajectories can be computed in three degrees of freedom about a spherical rotating earth. Four powered stages and three upper stage thrust cycles can be accommodated. Coast periods are permitted between each stage. Aerodynamic lift and drag forces are included in the computation. The optimization routine simultaneously considered the launch direction and time the interstage coast durations and the upper stage thrust sequencing, the complete pitch and yaw attitude histories and terminal constraints. Immediate constraints may be introduced on angle attacks, coast orbit perigee altitude or on the product on angle attack and dynamic pressure. The closed loop procedure greatly facilitates the satisfaction of terminal constraints and reduces the number of iterations required to achieve convergence.

PRICE: A Program for Improved Cost Estimation.

The PRICE computer program was developed in order to rapidly generate preliminary estimates of total program costs for mission studies of V-STOL and conventional transport aircraft, hypersonic aircraft and reusable space transportation system. The program uses cost estimating relationships based on historical cost data for conventional and advanced aircraft, spacecraft and launch vehicles. The approach is based on correlating cost with parameters such as weight and thrust, then adjusting the results with complexity factors to account for differences in material and type of construction, performance level, etc.

REPORT: Report Generator.

REPORT is a executive program (DLG) option which allows the user to interrogate the data base and format engineering reports. Descriptive information may be provided in any format with delimited data base names inserted where data base requests are desired. The report is automatically printed with descriptive information printed exactly as originally specified and delimited data base names replaced by the corresponding data base information.

SBOOM: Sonic Boom Prediction for Shuttle Type Vehicles.

The SBOOM program is based on the prediction technique of Thomas (references). The prediction technique calculates the far-field overpressure from the near-field pressure signatures measured in the wind tunnel. The wind tunnel results generally are stored in a data base and accessed by the computer program for interpolation/extrapolation based on geometric similarity of the pressure signatures. Wind tunnel data in the data base was generated for space shuttle type configurations. The approach used in determining ground overpressure is to describe the wave form of the sonic boom wave by several wave form parameters and then obtaining equations for the parameters as function time. This approach has the advantage of being simple and providing a convenient method for extrapolating experimental signatures because the signatures are dealt with directly. The input consists of the vehicle conditions and the environment in which the vehicle is operating as a basic condition operating the program. Many flight conditions can be equated through the normal input channels or the program will accept flight condition data from trajectory generation programs stored on an auxiliary file.

SEPARTE: The Program to Simulate Separating Stages of Launch Vehicles.

The program is a twelve-degree-of-freedom separation program for studying space shuttle separation problems. Each stage of the separating vehicle is represented by six degrees of freedom. Aerodynamic data governing the motion of the separating vehicle is staged into the program from a storage file because of the large bulk of data. The program generates Gerber plots of the separating vehicle components at specified time intervals. Separating components are represented by simplified geometric shapes.

SKINF: Turbulance Skin Friction Drag Program.

The program computes the skin friction drag of a vehicle including the effects of distributed roughness and temperature of the surfaces at arbitrary combinations of Mach number and altitude. Calculations can be made using either the standard day or the +10 degrees hot day atmospheres. Input consists of flight conditions (Mach-altitude), the wetted areas, reference length and form factors for all of the components of the aircraft in the mean roughness height and emittance of the surfaces. Wetted areas in reference lengths may be obtained from the program WETED.

SSAM: Swept Strip Aeroelastic Model.

The program performs an aeroelastic evaluation of the wing span-wise flight load analysis including the complete aircraft balance for a specified set of steady state maneuvers and/or design lift conditions. Here, proper inclusion of the wing-body and the nacelle aerodynamic and weight effects are included in order to compute the required balance tail load which is reflected in the wing load calculation. The flight loads, including the aerodynamic and wing dead weight loads, are converted into structural wing box bending and torsion loads to evaluate the resulting bending and torsional stresses. If the calculated wing stress exceeds the allowable wing stresses, a new set of values of wing section stiffness values are selected to match the allowable stress distribution specified within the program data. The wing aeroelastic load solution is then repeated until the calculated and allowable wing stresses are matched. The cycling process is fast and usually requires three to five cycles to converge depending upon the error margin set within the program. The final calculation is the wing box weight based on the final set of EI and GJ values obtained. The analysis is limited to subsonic flight conditions.

SSSP: Space Shuttle Synthesis Program.

The program automates the trajectory weights and performance computation essential to predesign of the space shuttle system for earth-to-orbit operation. The two-stage space shuttle system is a completely reusable space transportation system, consisting of a booster and an orbiter element. The SSSP major subprograms are detailed weights/volume routine, a precision three dimensional trajectory simulation and the iteration and synthesis logic necessary to satisfy the hardware and trajectory constraints. Three versions of SSSP are available representing some early space shuttle concepts in the predesign stage of the shuttle project.

TOLAND: Take-Off and Landing Program.

The program was constructed by NASA's Advanced Concepts and Mission Division, OART. The program provides simplified high lift aerodynamics based on DATCOM methods, a ground roll analysis, rotation logic and climbout to clear a fifty-foot obstacle. Angle-of-attack in the ground run and rotation maneuvers are determined from the vehicle geometry which is input to the program.

TOP: Trajectory Optimization Program.

A steepest-ascent optimization program has been developed which is capable of optimizing the flight path of a wide class of vehicles. The program will optimize rockets, air-augmented rockets, ramjets, scramjets, turbojets and glide vehicles. Unique features are incorporated which allow extension of optimization procedures into the airbreathing propulsion field, particularly to supersonic transport type vehicles.

The body pitch angle is used for in-plane trajectory shaping in place of the usual angle-of-attack control variable. Additional control variables include bank angle, engine throttling and a variable geometry control variable for vehicles with variable sweep wings, drag brakes, etc. Enroute placards are available to allow optimization within realistic constraints, such as control limits, structural loads, engine operating limits, manned vehicle requirements and geopolitical limitations such as sonic boom.

The optimization is accomplished with an automatic step-size controller and with automatic control variable weighting matrices to allow problem solution in a single computer run. Automatic plotting capability is included. Multistaged vehicles and problems involving variable initial conditions may be optimized. A variable step Runge-Kutta integrator performs the derivative evaluations.

TOPLOT: Plot Generator for TOP.

This program interrogates the output file from TOP (Boeing Trajectory Optimization Program) and generates time history plots of various performance and constraints functions generated during a top run. The output is placed on temporary disk and can be transferred to a physical tape for plotting using the PLOTSV procedure.

TREND: Subsonic/Supersonic/Hypersonic Aerodynamic Trade-Off Program.

The program provides rapid aerodynamic lift, drag and moment estimates and the subsonic, supersonic and hypersonic lift regimes. It is primarily designed to estimate high lift/drag re-entry vehicle aerodynamic characteristic, but the class of vehicles which may be analyzed by the program is of greater range than the primary class of vehicle. Some program modification may be required if the extension is too great. The program may be used for generating basic aerodynamic coefficients or it may be used for trending from known aerodynamic characteristics based on theoretical changes in the geometry. In the hypersonic flight regime, the program contains an optional aerodynamic heating computation capability. The program does not possess a transonic aerodynamic characteristic estimation capability.

VAMP: Volume, Area and Mass Properties.

The VAMP computer program calculates the mass properties, c.g. location, enclosed volume, wetted area and planform area of any closed structure that has a plane of symmetry. The vehicle is described to the computer program by ordered sets of X, Y, Z coordinates of points on its surface. The X, Y, Z coordinates are converted to quadrilateral elements for analysis. The mass properties of each quadrilateral may be computed from a thickness and density input for each quadrilateral or from a weight per unit area input at each point or from a combination of both.

The computed mass property totals may contain not only the contribution from the distributed mass on the vehicle surface wall, but additional masses may be added as "black boxes" by specifying each one's c.g. location and mass properties.

Program VAMP can also produce picture-like images of the vehicle or individual sections from the quadrilateral element data. This facilitates checking the input and visualizing and/or modified configurations. Orthographic, perspective and stereo views may be obtained.

VSAC: Vehicle Synthesis for High Speed Aircraft.

The VSAC program was developed for use in preliminary sizing studies of high speed flight vehicles. Single-stage aircraft with either takeoff and landing capability or airborne start, and two-stage vehicles consisting of an aircraft with an expendable, rocket-propelled booster can be sized. The single-stage aircraft are capable of either rocket or airbreathing propulsion and flight speeds to Mach 6. Two-stage configurations cover flight regimes to Mach 15, although no supersonic-combustion engines were considered.

Mission modeling is provided in a modular fashion to allow the program user maximum flexibility, and to allow improvement or substitution of subroutines as new studies may require. Analytic performance equations and numerical trajectory integration are provided, together with appropriate iteration routines.

The primary aerodynamics method is a component buildup procedure that requires the user to input only a geometrical description of the external vehicle characteristics and generates curve-fit expressions for the lift and drag coefficients. Stability and control are not considered.

WAATS: Weights Analysis for
Advanced Transportation Systems.

WAATS is a weights analysis program which uses the component buildup technique for weight estimation. Each component weight is based on the weight of the same component of similar vehicles that have actually been built or at least designed in great detail. The similarity law that gives the best correlation for most subsystems has been shown to be the power law formula:

$$y = a x^b.$$

The program logic assumes the propellant weight and physical characteristics are known. It performs the weight estimations with user supplied correlation parameters (a and b), estimated gross weight and estimated landing weight. An internal iteration loop cycles through the equations until convergence on weight is achieved.

WDRAG: Zero-Lift Wave Drag Program.

The program calculates the zero-lift wave drag at supersonic speeds for airplanes having arbitrary combinations of fuselage, wing, nacelle, horizontal fins and vertical tail. The input geometry description is in the format of CONPLOT and WETTED programs. The program may also be used to calculate (for any desired Mach number) the fuselage area distributions that are required for a minimum wave drag. Simplified fuselage constraints may be included for the above calculation. The fuselage may be cambered and may have arbitrary cross-sectional shapes. The wing may be twisted and cambered. The theoretical approach is based on the far-field linear theory that considered the relationship between the forces on the airplane and the momentum transported through the boundaries of a control surface completely surrounding the airplane.

WETTED: Wetted Area in Reference Length Program.

This program computes the surface wetted area and reference length for all components of a configuration based on corner-point geometry inputs to the program. The configuration may be arbitrarily divided into components on the fuselage and all aerodynamic surfaces. The wing may be divided into many stream-wise panels in order to approximate a strip integration for skin friction calculations. Input to the program is identical to CONPLOT. Output is suitable for use in the SKINF program.

LIBRARY ADDITIONS AND MODIFICATIONS

During the contract period a number of new programs were developed. In addition, several existing programs were modified for incorporation into the EDIN system. These additions and modifications dealt with the EDIN executive, geometry, graphics and general utilities software. Figure 9 summarizes some of the EDIN software which was developed or modified for the EDIN system.

The following sections provide a description of the newly developed or modified software. The description includes a program description, physical characteristics of the software and program usage requirements containing descriptions on the control cards necessary to execute program input and output. Among the many programs which were developed or modified during the contract period, only those which appeared to have been major developments are described in the following sections.

NEW PROGRAMS OR PROGRAMS WITH MAJOR MODIFICATIONS

EXECUTIVE

DLG	DATA MANAGEMENT PROGRAM.
DATA BASE	NAME ORIENTED DATA BASE.

GEOMETRY

GTM	GEOMETRY TECHNOLOGY MODULE.
PANEL	PROGRAM FOR PANELLING AEROSPACE VEHICLES.
SFIT	THREE-DIMENSIONAL SURFACE FIT.

GRAPHICS

PLOTTT	INDEPENDENT PLOT PROGRAM.
IMAGE	PICTURE DRAWING PROGRAM.
CIPHER	REPORT WRITING PROGRAM.

UTILITIES

	TECHNOLOGY MODULETTES.
TANK	FULL AND OXIDIZER TANK DESIGN PROGRAM.
SIZER	STAGE SIZING FOR MULTISTAGED VEHICLE.
VL70	READ AND INTERPOLATE DATA FROM SHUTTLE AERO DATA TAPE.
RDPRO	READS TRAJECTORY DATA TAPE.
WAB	WEIGHT AND BALANCE PROGRAM.
HDG	HEADING PROGRAM.

FORTRAN CALLABLE INTERFACE SOFTWARE

IF EXEC 8 PROCESSOR INTERFACE:



DMAN	EDIN DATA BASE INTERFACE. (DIRECT STORAGE AND RETRIEVAL)
ADD ---	EDIN DATA BASE INTERFACE (STORAGE WILL BE THROUGH DLG)
IF.	EXEC 8 PROCESSOR INTERFACE (COMMUNICATION WITH FILE ELEMENTS)

FIGURE 9 NEW OR MODIFIED SOFTWARE DEVELOPMENTS.

AIRFOIL: AIRFOIL GENERATION PROGRAM

The AIRFOIL Program has been modified to write a file of data in BCD format suited for the GTM to read. See reference 33 for GTM input requirements. The modification to AIRFOIL is in the form of a new subroutine, GAMOUT, which is called from the main program, FOIL. This subroutine writes a file of data on unit 4 as illustrated in figure 10. The data for each airfoil is written in two segments, defined in the x-z plane but written in generalized x-y-z point triplets.

The first segment contains the upper coordinates and the second segment contains the lower coordinates. There are as many pairs of segments written in unit 4 as there are input airfoils. See reference 23 for AIRFOIL input. Each segment of data on unit 4 is headed by a GTM formatted input header card record as follows:

BINARY INPUT	NACA	<table><tr><td>AIRFOIL</td><td>UPPER OR</td></tr><tr><td>NUMBER</td><td>LOWER</td></tr></table>	AIRFOIL	UPPER OR	NUMBER	LOWER
AIRFOIL	UPPER OR					
NUMBER	LOWER					
						
GTM COMMAND		GTM DATA BASE NAME				

Each segment of data is terminated by a separate card record containing a GTM terminates command:

END INPUT



BINARY INPUT	NACA	24012	UPPER
.00000	0.0	.00000	
-.01986	0.0	.63335	
.04829	0.0	.90805	
.13143	0.0	1.12356	
.22252	0.0	1.30831	
.31880	0.0	1.47330	
.41885	0.0	1.62416	
.52182	0.0	1.76424	
.62713	0.0	1.89573	
<hr/>			
			
96.14205	0.0	.75744	
98.13502	0.0	.43465	
100.00000	0.0	.00000	
END INPUT			
BINARY INPUT	NACA	24012	LOWER
.00000	0.0	.00000	
.26986	0.0	-.57302	
.45171	0.0	-.78797	
.61857	0.0	-.94430	
.77748	0.0	-1.07044	
.93120	0.0	-1.17738	
<hr/>			
			
82.06307	0.0	-1.91959	
84.06904	0.0	-1.74173	
86.07515	0.0	-1.55843	
88.08140	0.0	-1.36961	
90.08780	0.0	-1.17513	
92.09435	0.0	-.97485	
94.10106	0.0	-.76858	
96.10793	0.0	-.55612	
98.11497	0.0	-.33723	
100.00000	0.0	.00000	
END INPUT			

FIGURE 10 ILLUSTRATION OF GTM INPUT FILE
GENERATED BY THE AIRFOIL PROGRAM.

CIPHER: REPORT WRITING PROGRAM

The CIPHER program is a general purpose report writer and documentation program. CIPHER is designed to take an input stream of characters which is a mixture of both the text to be output and embedded control commands. The program processes the actual text according to the instructions given and produces a stylized report. The following capabilities are available:

- User Specified Formats.
- Upper and Lower Case.
- Greek Symbols.
- Indexing and Page Numbering.
- Cross Referencing.
- Superscripts and Subscripts.
- Footnotes.
- Equations.
- Math Symbols.
- Mixed Character Height.
- Automatic Tab Setting and Paragraphing.
- Automatic or User Controlled Paging.
- Carriage Control.

CIPHER is interfaced to the Tektronix 4012 and the MOPS terminal systems. By plotting the characters, any font may be generated as well as any symbol which can be described. Both upper and lower case English alphabetical characters along with most of the Greek alphabet and numerous special mathematical symbols are available.

Commands begin with a \$ symbol followed by one (1) or more English characters. About 20 commands to perform virtually any layout function are available.

Physical Characteristics

HOST COMPUTER	UNIVAC 1110 AND CDC 6600
FILE NAME(S)	EX42-00002*CIPHER (ARS)
ABSOLUTE ELEMENT NAME	CIPHER
LANGUAGE	FORTTRAN
PROGRAM SIZE	27000 DECIMAL

CARD SOURCE

2100

OPERATING MODE

DEMAND

DISPLAY INTERFACE

TEKTRONIX, MOPS

Program Usage

Input description presented below were generated using the CIPHER program.

The characters which are available represent most of the Alpha-numeric, Greek, and Mathematical Operators commonly required. The program is designed to accept new characters so that new or special characters required can be easily added.

ALPHABETIC CHARACTERS

<u>Character</u>	<u>Input symbol</u>	
a,A	A	Note: see Carriage Control
b,B	B	
c,C	C	
d,D	D	
e,E	E	
f,F	F	
g,G	G	
h,H	H	
i,I	I	
j,J	J	
k,K	K	
l,L	L	
m,M	M	
n,N	N	
o,O	O	
p,P	P	
q,Q	Q	
r,R	R	
s,S	S	
t,T	T	
u,U	U	
v,V	V	
w,W	W	
x,X	X	
y,Y	Y	
z,Z	Z	

ORIGINAL PAGE IS
OF POOR QUALITY

GREEK CHARACTERS

Character	Input symbol	name
α	$\$GA$	ALPHA
β	$\$GB$	BETA
Γ	$\$G\Gamma$	CAP GAMMA
γ	$\$GG$	GAMMA
Δ	$\$G\Delta$	CAP DELTA
δ	$\$GD$	DELTA
ϵ	$\$EPSILON$	EPSILON
ζ	$\$ZETA$	ZETA
η	$\$ETA$	ETA
Θ	$\$1THETA$	CAP THETA
θ	$\$THETA$	THETA
λ	$\$GL$	LAMBDA
μ	$\$MU$	MU
ν	$\$NU$	NU
ξ	$\$XI$	XI
Ξ	$\$1XI$	CAP XI
π	$\$PI$	PI
ρ	$\$RHO$	RHO
Σ	$\$SUM$	CAP SIGMA
σ	$\$SIG$	SIG
τ	$\$TAU$	TAU
υ	$\$GV$	UPSILON
ϕ	$\$PHI$	PHI
Φ	$\$1PHI$	CAP PHI
χ	$\$GX$	CHI
Ψ	$\$1PSI$	CAP PSI
ψ	$\$PSI$	PSI
Ω	$\$G\Omega$	CAP OMEGA
ω	$\$GO$	OMEGA

MATHEMATICS OPERATORS

<u>OPERATOR</u>	<u>INPUT SYMBOL</u>
-----------------	---------------------

+	+
-	-
±	\$±
*	*
/	/
≠	\$NE
<	\$LT
>	\$GT
≤	\$LE
≥	\$GE
=	=
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

PUNCTUATION AND MISCELLANEOUS OPERATORS

<u>OPERATOR</u>	<u>INPUT SYMBOL</u>
-----------------	---------------------

((
))
[[
]]
:	:
'	\$APOS
"	\$OQ
\$	\$DOL
?	\$Q
!	\$EX
"	\$CQ
{	\$OBRC
}	\$CBRC

These characters represent the list of available characters. Any characters desired which are not in the list can be easily added.

The functions listed below are special functions and operators provided to allow the user to construct a readable and attractive document. Some of these functions duplicate typewriter characteristics. Others are required to construct equations. Some are directed to Manual construction. The purpose of these operators and functions is to give the user firm control over his documents form and construction without requiring a pre-existing copy.

OPERATOR	DESCRIPTION
----------	-------------

\$NL

\$DS

These functions are "Carriage Control" characters. \$NL insures that the next character begins on a new line. \$DS insures that the next character begins on a new line but one line down. In other words this is the carriage return. These characters are used only when it must be insured that information begins printing on a new line. Routine carriage returns are handled automatically.

##

Information is supplied to this program in EBC format. The card images may come from INPUT or CDC UPDATE. In either case, only the first 72 columns may be used. Since English is not a "regular" language, no grammatical structure can be implied. The entire grammatical structure must be card input. This means that every card input character--including blanks--must be considered. The ## character is the "End Of Information" symbol. No information on a card will be considered after a ## has been encountered.

examples:

card 2 LAMB. \$NLTHIS FLEECE ...##

card 1 MARY HAD A LITTLE ##

These cards would produce:

Mary had a little lamb.

His fleece...

#Hi

This character or function changes the character size. It applies to all characters following until the height is changed again. The following series of characters represents the character sizes for i=0,1,2,3,4,5,6,7,8,9.

0,1,2,3,4,5,6,7,8,9,

#PAGE

This function insures that the information following begins on a new page. When this function is encountered, a major information change is assumed. #TAB is set to zero. Normally a new page is started when the present page is full. In this case no changes are made to any of the existing operating parameter.

#PP

This function is the "Paragraph" symbol. Information following this function will begin on a new line with a blank line between the lines. #TAB is set to zero.

↑
↓
↑↑
↓↓

These functional characters are equivalent to the "Shift" key on a typewriter. The ↑'s are the "upper case" shift. The ↓' are the "lower case" shifts. The single arrows apply only to the following character. The double arrow characters apply to all following characters until the next double arrow functional is encountered. Examples--each card image is followed by the printed line generated.

↑↑MARY↓↓ THE IVAMP OF ISAVANNAH#EX
MARY the Vamp of Savannah↓
↑↑MARY↓↓ THE IVAMP OF ISAVANNAH#EX
MaRy the Vamp pf Savannah↓

#UL

#UL(-----)

This function is the underline function. The first form will cause an underline to be drawn from the #UL to the

next space. The second form will underline the entire expression enclosed within the parenthesis. NOTE: The parenthesis will not be included in the printed expression.

Examples:

{ULUNDERLINE TESTS AND #ULEXAMPLES}
underline tests an examples
{UL(UNDERLINE TESTS AND EXAMPLES)}
underline tests and examples

\$TAB1

This function will cause an indentation of all subsequent lines. The "i" parameter determines the amount based on the equation:

$$\text{Indentation}_{\text{inches}} = i * 0.5$$

This indentation is measured from the left margin. This function assumes a major information change. Upon encountering this function the preceeding line is printed and a new line is started.

\$T1

This function is used to assemble tabular data. An indentation for the next character is computed using:

$$\text{Indentation}_{\text{inches}} = i * 0.5$$

This indentation is measured from the LEFT MARGIN + TABSETTING. This function may be used within any other function such as UL, TITLE, etc.

\$SEE(name)

\$REF(name)

These two functions represent the "Cross Referencing" capability in this report generator. "name" can be any series of up to ten numbers and letters. The \$REF function declares that this point in a document may be referenced by other sections of the same document. The \$SEE function cause a comment to be injected into the document referensing this section. The following example demonstrates this capability.

{REF(THISSECTION)}
{SEE(THISSECTION)}

#TITLE1(....)

This function will cause the expression within the parenthesis to be centered. The character height of the title can be controlled by the "i" parameter. The values of "i" may vary from 0 to 9. The height sequence is the same as that given in the #Hi function description. This parameter is optional. If omitted, the title height will be that of the previous output.

Example:

#TITLE5(TITLE DEMONSTRATION)

TITLE DEMONSTRATION

#SUP(....)

#SUB(....)

These functions accomplish the subscripting and superscripting required in technical reports. The parenthesis in these function are optional. They are required for multi-character subscripts and where nesting is required for subscripts to superscripts, etc. These functions always refer to the previous character. These functions are also used for the upper and lower limits to an integral sign.

Example 1. A single character subscript

#SIN#6B#SUBT

$\sin \theta_t$

Example 2. A multi-character subscript

#SIN#6B#SUB(TIME)

$\sin \theta_{time}$

Example 3. Superscripts with sub-and-superscripts

#SIN#6A#SUP(#GT#SUP2#SUBP)

$\sin \theta_p^{>2}$

#INDEX(....)

This function accomplishes the indexing. The expression within the parenthesis will appear in the index. The expression may be up to 50 characters long, and may contain any characters or functions. The alphabetization is based on the first 20 characters only. Subscripts and Superscripts are taken into account for alphabetization.

Example 1. This card is the Index Generator for this

INDEX(INDEX GENERATION)

These functions are all variable height characters. Most of these are sometimes used in nested situations. The purpose of these functions is to add clarity to a nested group of these symbols. "I" is the standard height sequence given in the #Hi section with one addition. "I" may be equal to T. In this case the character will be triple normal height. This is to be used when a numerator-denominator expression is to be enclosed in symbols.

\$T \$0 \$1 \$2 \$3 \$4 \$5 \$6 \$7 \$8 \$9

$$\langle \quad \quad \quad \rangle$$

An integral sign at times requires upper and lower limits. In this case the upper and lower limits are included within the parenthesis as superscripts and subscripts. "i" is the optimal height key as described in the preceding section.

$$\int_{-\infty}^{+\infty} (A+B) dx$$

$$\int_{-a}^{+a} (a+b) dx$$

This function will draw a card image over the expression within the parenthesis. All the information contained within the parenthesis will appear on one line.

⌘CARD(CARD IMAGE EXAMPLE)

card image example

\$NUM(.....)

\$DEN(.....)

These functions allow an numerator/denominator expression to be written with the numerator directly over the denominator. Either may appear first, however no other expressions can appear between them.

Numerator and denominator expressions may not be nested. The longer of the two expressions should come first. If is done the second expression will be centered with respect to the first.

Example:

$$\frac{\$DEN(A+C*D)\$NUM(2\$PI)}{2\pi}$$

$a+c*d$

\$SQRT(....)

This function draws the Square Root symbol over an expression. The expression may be multi-line as in a numerator/denominator expression.

Example 1.

$$\sqrt{\$SQRT(A+B \$SUP3)}$$

$\sqrt{a+b^3}$

Example 2.

$$\sqrt{\$SQRT(\$DEN((A+B))\$NUM(E\$SUP2))}$$

$\sqrt{\frac{e^2}{(a+b)}}$

Example 3.

$$\frac{1}{\sqrt{\$DEN(\$SQRT((A+B)\$SUP3))\$NUM(1)}}$$

$\frac{1}{\sqrt{(a+b)^3}}$

\$CODE

\$ENDCODE

These functions are used to list FORTRAN or other code. The distinction is that on line of card input occupys one line in the report. NOTE: The || must not be used within the code block. The first symbol signals the beginning of a code block. The second symbol signals the end of a code block.

PROGRAM EXAMPLE(INPUT,OUTPUT)

DIMENSION X(10)

DO 10 I=1,10

COT PROCESSOR

Due to the manner in which images are stored upon mass storage, the presence and sequence numbers in columns 73-80 significantly increases the physical amount of storage required. Blanking these columns is a difficult and enormous task via the ED processor, therefore, a small processor called COT has been developed for this purpose.

This processor and its two associated SSG run streams can do this blanking quickly and conveniently. The processor has two modes of operation of which one and only one must be specified.

The "T" option will truncate the named element and the "C" option will compress it.

To use the processor to compress the element FROG in file HOP, one simply types:

```
@EX42-00002*UR.COT,C HOP.FROG,HOP
```

To truncate it the call is the same except that the "T" option is used.

There are also two SSG skeletons in UR that may be used to compress or truncate all symbolic elements of a file. To use these, simply type:

```
@USE FN,your-file-name  
@ADD UR.TRUNCATE
```

or

```
@USE FN,your-file-name  
@ADD UR.COMPRESS
```

To "expand" a compressed element, one needs to type:

```
@ELT,IGN HOP.FROG  
@ADD HOP.FROG
```

Note that the ELT processor must be used and that both the "I" and the "G" options are needed hence the @ADD behind the ELT call.

By compressing all symbolic elements in a file savings in required storage space has been observed as high as 50% with mean near 25%.

FLOWGEN: AUTOMATIC FLOW CHART GENERATOR PROGRAM

This section describes the automatic flow charting program, FLOWGEN, of reference 36. FLOWGEN produces flow charts of Fortran language subroutines and is available for use on the Univac 1100 series computing system. FLOWGEN is a CALCOMP proprietary program which has been purchased for use on the Johnson Spacecraft Center system.

Program Control

The FLOWGEN program is on the Univac 1100 mass storage file:

```
FM9*FLOWGEN
```

The use of the program requires the generation of an absolute element based on the type of output device desired.

```
7@MAP,N FM9*FLOWGEN.FLOWGC,TPF$.FLOWGC (for CALCOMP)
```

or

```
7@MAP,N FM9*FLOWGEN.FLOWGM,TPF$.FLOWGM (for Microfilm)
```

For CALCOMP plots, an output tape must be assigned by

```
7@ASG,T 19,8C,CCP
```

The automatic flow chart processor, FLOWGEN, has been used to produce the computer flowcharts for the EDIN system. An illustration of its use is shown in the following Exec 8 run stream:

```
@ERS
@COPY,S DLG.,TPF$.
@MAP,N FM9*FLOWGEN.FLOWGM,TPF$.
@XQT
  INPUT
  IREAD=1,ISPEC=2
$END
```

Program Input

The various flags which must be specified by the user are input by means of data cards sent to a namelist like input. The first card of this input is a card starting in card column 1 and containing the word INPUT. This card is a required card and follows the XQT card. The NAMELIST input is terminated when a card containing the word \$END is encountered.

The NAMELIST input cards are free field cards of the form:

NAME=VALUE

The various NAMELIST parameters with their meanings follow:

IREAD	Subroutine Selection Flag. = 0, process only those routines specified by input. = 1, process all the Fortran elements on the specified files (preset to 0)
ISPEC	Specification Statement Inclusion Flag. = 1, do not include specification statements in the flow charts produced. = 2, include the specification statements (preset to 1).
KRPCF	PCF Tape Unit Number. = 0, for drum processing. = 5, for card reader input. = N, for PCF tape unit N (preset to 27, = Unit X)
NFILES	The Number of Files to be Processed on Logical Unit KRPCF (preset to 1).
NSTART	Number of the File on Unit KRPCF at which to begin Processing (preset to 1).
REDUCE	CALCOMP Plot Size Reduction Flag. = N, reduce the size of the plots by N percent.

GTM: GEOMETRY TECHNOLOGY MODULE

The Geometry Technology Module (GTM) is a system of computerized elements residing in the EDIN (Engineering Design and Integration) System library developed for the generation, manipulation, display, computation of mass properties and data base management of panelled geometry. The GTM is composed of computer programs and associated data for performing configuration analysis on geometric shapes. The program can be operated in batch or demand mode and is designed for interactive use. The significant features of the program are:

1. Data bases containing two and three dimensional shapes including standardized shapes generated by the GTM.
2. An executive computer program containing a user orientated language for controlling the generation, display and calculation of mass properties on selected vehicle components.
3. An auxiliary computer program and data base for the construction and storage of language elements, menus, user instructions and messages.
4. A library of independent geometry generation programs for the creation of specialized geometric panelling.

The basic capabilities exhibited by the GTM are summarized in figure 11. The first Geometry Definition illustrates the user capability to generate new geometry, use existing geometry in standardized formats and use externally generated geometry from another geometry generation program. In conjunction with the user ability to define geometry, GTM provides the user with the capability to manipulate geometry. The manipulation includes translations, rotations, scaling, merging of geometric components, division of geometry and surface fits. GTM also provides for display of geometry. The geometric display can be translated, rotated, overlayed or zoomed for image enhancement. Mass property evaluations may be commanded. The evaluation includes a printout of weights, volume, center of gravity and surface areas of the accessed geometry. The GTM also provides the capability for easily interfacing with the EDIN system.

Program Description

The executive GTM module is composed of several major executive levels. These levels are called by the GTM executive, named MASTER. The major executive levels are the input module, cluster edit module and segment edit module. Figure 12 illustrates the GTM program organization.

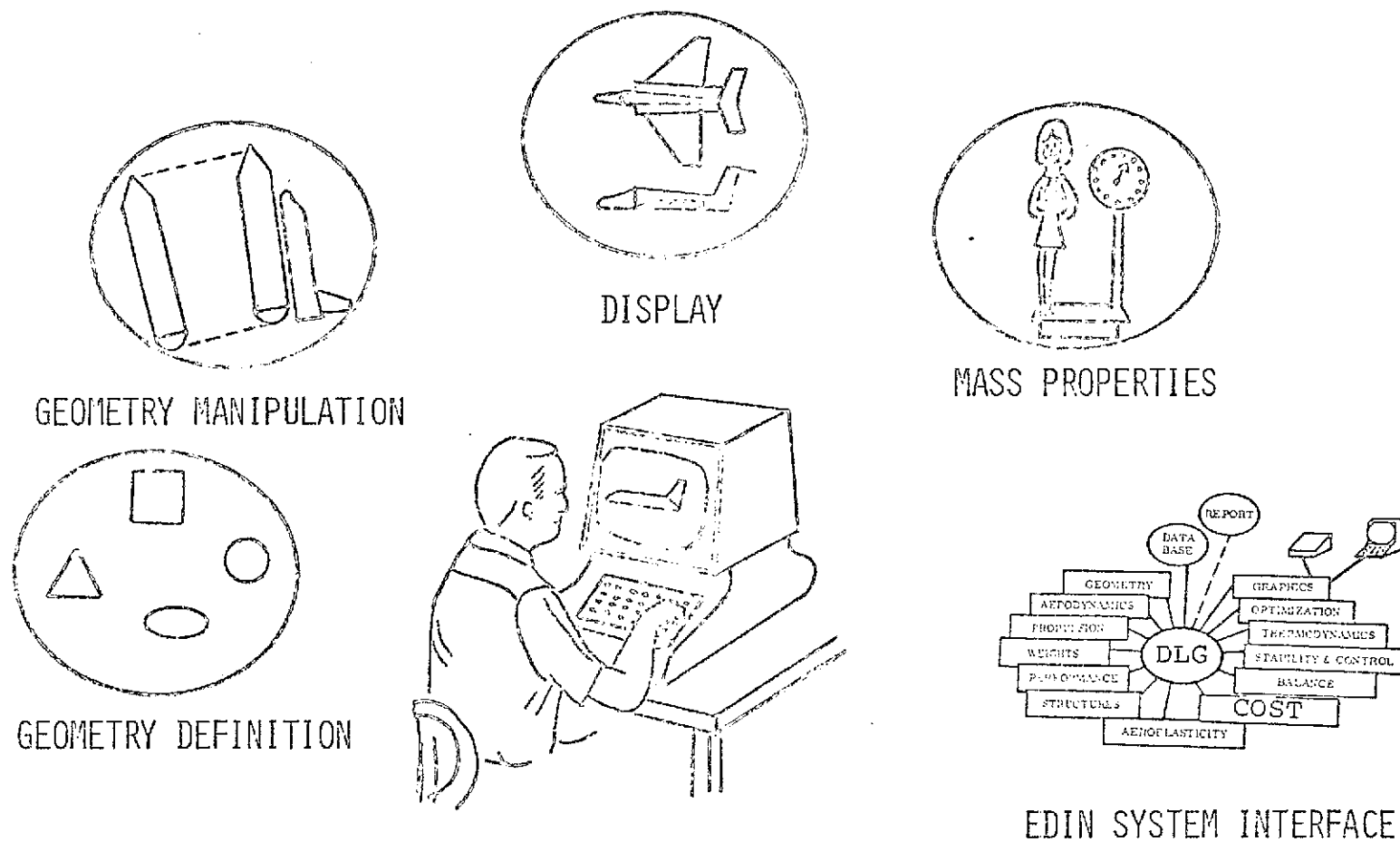


FIGURE 11 GTM CAPABILITIES.

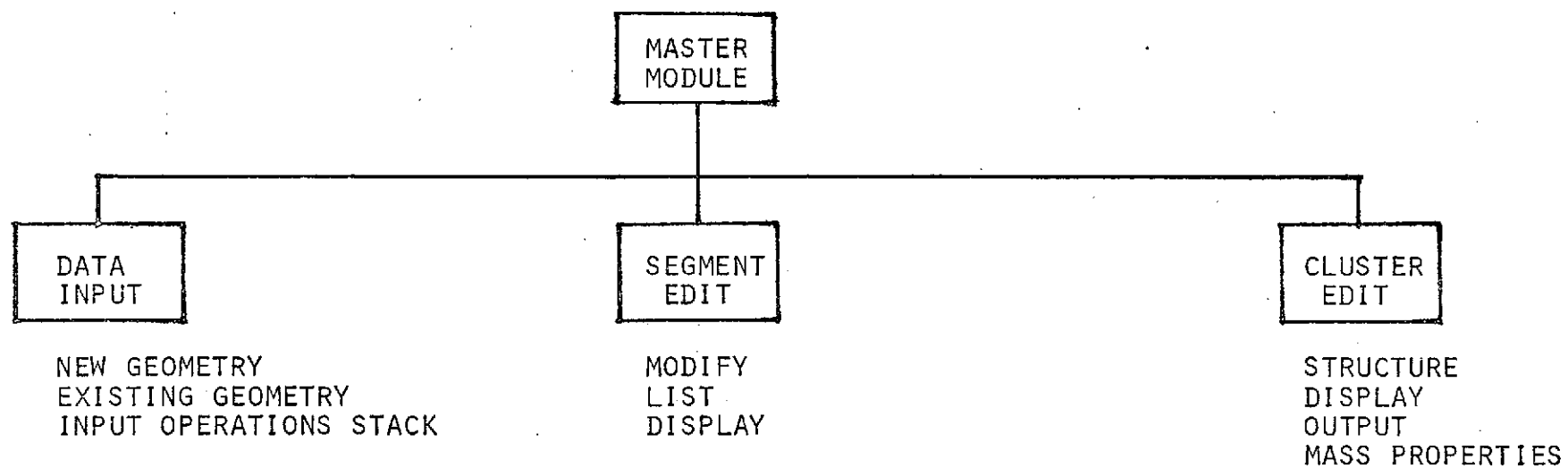


FIGURE 12 GTM PROGRAM ORGANIZATION.

The MASTER module (GTM Executive) is the control point in the GTM from which all sublevel executives are accessed. It contains its own language set which allows the user to perform data base management functions, access sublevel executives and general program control. Three primary sublevel languages are available, input, segment edit and cluster.

The input sublevel executive is provided for reading data which is stored in specific geometry formats. Two are available, the Gentry format of reference 31 and the GTM format. GTM format allows free-field data to be entered. The data may be any type of information. This data is read in and stored in the data base geometry tree structure. The INPUT module contains its own language set and associated menus, which can be displayed upon command.

The CLUSTER EDIT Module contains a language subset and instructions necessary for creating and maintaining the geometric data tree structure. Functions are also provided for translation, rotation and scaling of tree stored data and output of the data in forms for interfacing with other EDIN technology modules. In addition, it contains the necessary logic to display geometry for image viewing. The display functions have a number of features which allow the user to zoom in on a specific region, overlay geometry, scale geometry and filter geometry for resolution. Mass properties evaluations are also commanded from the CLUSTER EDIT Module.

The SEGMENT EDIT Module provides the capability to compose geometric shapes, manipulate geometry at the segment level and display of geometric segments. Specific operations include translation, rotations, scaling, point redistributions, segment cutting, point edit commands and display. The module contains its own language subset addressable by the user.

The GTM provides the capability of maintaining and updating geometry information in a name oriented data base. The geometry can be a section, component or a cluster.

A section is defined as a sequence of arbitrary X,Y,Z points defining a line in three dimensional space. A component is a collection of sections approximating surface points in three dimensional space. A cluster is defined as a collection of components which form a complete or partial surface configuration. The data can be tree structured at the cluster level so analysis can be performed on groups or collections of data with relatively simple data structure definitions. Once the data is assessed by the GTM, a variety of manipulation techniques are available at all data definition levels through the GTM language.

When geometry is inserted into the data base in a tree structure, the tree can be defined at four levels as illustrated by the following:

CLUSTER NAME
COMPONENT NAME
SECTION NAME
DATA POINT

Individual geometry data sets are generally stored only once though each set may belong to more than one cluster (tree).

The GTM has advanced to the state where geometry can be stored by name at several hierarchical levels in a tree structure. Editing of the data can be performed at all levels of data definition. The program contains the basic utilities which permit the development of user orientated manipulation of geometry and critical manipulation functions such as scaling, rotation and translations.

Additional information on the GTM program descriptions may be obtained from reference 33.

Physical Characteristics

The following presents a summary of the physical characteristics of the GTM:

HOST COMPUTER:	Univac 1110
FILE NAME(S):	EX42-00002*GTM2. (SOURCE/ RELOCATABLES) EX42-00002*GTM. (ABSOLUTE ELEMENT) EX42-00002*DATA5. (DATA BASE/ LANGUAGE) EX42-00002*ODIN-DBINIT2. (MAP ELEMENT)
ABSOLUTE ELEMENT NAME:	GTM
LANGUAGE:	FORTRAN V
PROGRAM SIZE:	24000 DECIMAL (OVERLAYED)
CARD SOURCE:	+ 12000
OPERATING MODE:	BATCH OR DEMAND
DISPLAY INTERFACE	TEKTRONIX

Program Usage

The computer program usage requirements described in this section are oriented toward the Univac Exec 8 1110 version and specifically towards the Johnson Spacecraft Center's installation. The actual program input (language commands) described are applicable wherever the program is installed but the control cards of the program will differ from computer to computer.

Control Cards

The control cards for execution of the GTM are illustrated by figure 13. After input of the run card, an assign of temporary file one (1) is required for data base storage. The data base and associated GTM language set presently resides on file EX42-00002*DATA5. Instructions for creating a new data base and the associated language structure are contained in reference 33. A copy of the DATA5 file to temporary file one (1) is required to protect the integrity of the data base. All I/O is stored and retrieved from file 1 during execution. If the user wishes to retain any data base entries during execution, he may permanently save these entries by a copy of file 1 to DATA5 prior to termination. Following the execute command, the user is free to enter GTM orientated commands. A brief discussion of these commands follows. Further information on the command structure and associated language set can be found in reference 33.

Program Input. - The GTM operates upon a data base of stored information which is manipulated by a language set which is the input to the program. The information which follows describes in brief the language structure of the GTM.

The GTM consists of a master level executive and other lower level executives. The executives are responsible for executing a specific task upon request by the user. A language consisting of manipulation commands for controlling the GTM at each executive level is available.

Master Level Language. - The following commands are the statements available in the master executive at the present time:

- *IMAGE INPUT
- *INPUT
- *CLUSTER EDIT
- *SEGMENT EDIT
- SAVE DATA BASE (___)
- OPS STACK (___:___:___)

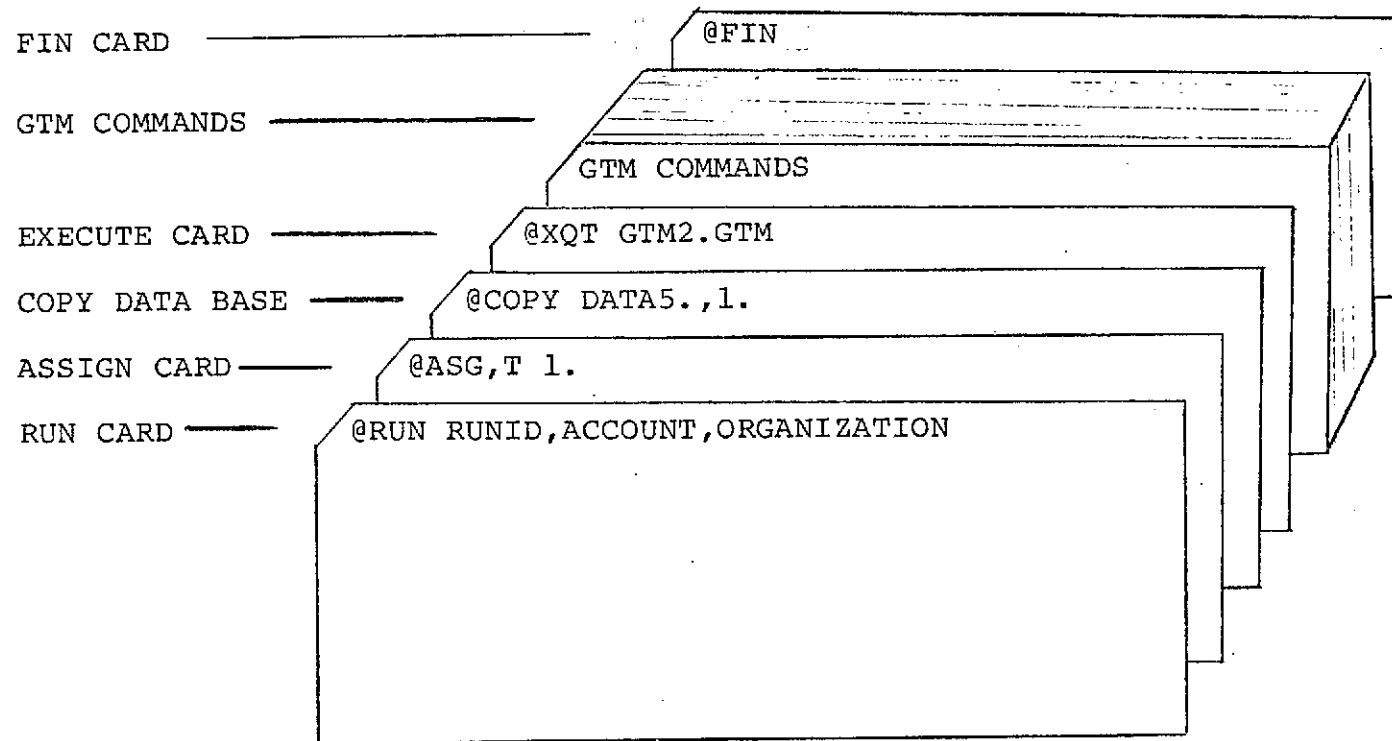


FIGURE 13 TYPICAL GTM RUN STREAM

MENU

EXIT

STOP

*Sublevel executives

Descriptions of the Commands

Image Input. - This command will cause a transfer to the Image Input Executive. The Image Executive is provided as a means of reading data which is stored in the arbitrary body coordinate (IMAGE Program) format of reference 27. This data is read in and stored in the data base geometry tree structure.

General Commands:

MENU - Provides a list of available commands at the current language level.

EXIT - Returns control to the master level language.

Data Source Commands:

DATA BASE (__:__:__) - The data resides in the data base in the card image form.

BCD FILE n _ The data is a file n and is formatted data.

BINARY FILE n - The data is a file n and is unformatted or binary data.

Tree Structuring Commands: The IMAGE formatted data contains status codes of 0,1,2,3. All points are considered status 0 except as follows:

Status 1 Beginning of a new section.

Status 2 Beginning of a component or subcomponent (synonymous in the GTM)

Status 3 End of a cluster of components.

The status flags are used when entering the data into the GTM data base to position the data within the geometry structure. For each Status 3, or the beginning of a file of input, the following command must be input:

[CLUSTER]
[VEHICLE] = __:__:__

This command will cause all subsequent data with status less than 3 be entered into the tree under the name __:__:__. For each Status 2 encountered, the following command must be input:

$$\left[\begin{array}{c} C \\ \text{COMPONENT} \end{array} \right] = _:_:_$$

This command will cause all subsequent data with Status 2 to be stored in the data base geometry tree structure under this name.

NOTE: For Status 1, the section names are set by default only.
The default names are:

SECTION 1

SECTION 2

;

;

SECTION n

for each Status 1 encountered in the component.

INPUT. - This command will cause a transfer to the free-field data and can be used to enter data blocks and any type of information. The input block has the following requirements:

HEADER This statement gives the type of storage and the name under which the data is to be entered into the data base.

END This statement signifies the end of the input block.

Types of Header Statements:

BCD INPUT (__:__:__)

This header is used to store the card image data in the data base.

NUMERIC INPUT (__:__:__)

This header is used to enter numeric data into the data base. The data is read in a free-field format. The values on the card must be separated by delimiters, which can be either a space or a comma. Commas back to back specify null fields between them.

OPS STACK ____:____:____

This statement is used to input an OPS Stack to the data base. An OPS Stack is an instruction string of commands which can be executed by using the OPS STACK ____:____:____ command.

Input items can be read from other files using the READ FILE command. This will cause the read to transfer to the specified file and continue with that file until the file is exhausted.

Cluster Edit

The Cluster Edit language subset contains instructions necessary for creating and maintaining the geometric data tree structure. Functions are also provided for translation, rotation, display and scaling of tree stored data and output of the data in forms for interfacing with other programs.

Edit Commands.

BUILD	[CLUSTER COMPONENT SECTION]	__:__:__
BUILD	[SEGMENT]	__:__:__
ACCESS	[CLUSTER COMPONENT SECTION SEGMENT]	__:__:__
LOCATE	[CLUSTER COMPONENT SECTION]	__:__:__
INSERT	[CLUSTER COMPONENT SECTION]	(__:__:__)
DELETE	[CLUSTER COMPONENT SECTION]	(__:__:__)
REPLACE	[CLUSTER COMPONENT SECTION]	(__:__:__)
COPY	[CLUSTER COMPONENT SECTION]	(__:__:__)
COPY	[SEGMENT]	(__:__:__)
ADD	[CLUSTER COMPONENT SECTION]	(__:__:__)

Output Commands.

[LIST
L] [CLUSTER
COMPONENT
SECTION] (__:__:__)

TREE LIST (__:__:__)

LIST AVAILABLE CLUSTERS

COPY BINARY (__:__:__)

COPY BCD (__:__:__)

Transformation Commands.

Rotation

[PSI
YAW] = ____

[THETA
PITCH] = ____

[PHI
ROLL] = ____

[RCEN
ROTATION CENTER] = __,'__,_

[RUEC
ROTATION VECTOR] = __,'__,_

[ROT
ROTATE]

Scaling

MAG = ____

XMAG = ____

YMAG = ____

ZMAG = ____

[MAGC
MAGNIFICATION CENTER] = __,'__,_

[SC
SCALE]

Display Commands.

DISPLAY ____:____:____
DISPLAY +____:____:____
DISPLAY -____:____:____
REFRESH
AFILT = ____
RFILT = ____
PSI = ____
THETA = ____
PHI = ____
SYM
NOSYM
SCALE = ____
ZOOM = ____, ____

Translation

XMOVE = ____

YMOVE = ____

ZMOVE = ____

MOVE

Bounding Commands

START ____:____:____

STOP ____:____:____

Register Commands

ZERO	[BUILD ACCESS LOCATE]
------	-----------------------------

ACCESS	[BUILD LOCATE]
--------	-------------------

BUILD	[ACCESS LOCATE]
-------	--------------------

LOCATE	[ACCESS BUILD]
--------	-------------------

Miscellaneous Commands

MENU

OMIT

EXIT

Description of Commands.

Edit Commands. Addressing a tree structure requires the maintenance of a list of data pointers, one for each level of the tree. These lists are called registers. Three registers are maintained in the GTM, a build, an access and a locate register. The build register is constructed by the GTM when the geometry is initially stored. The build register can be thought of as the output register. For instance, data is copied from the access register to

the build register. The access register is used when a geometric manipulation is performed. The access register can be thought of as the input register, although many commands affect the data in this register. The locate register is a temporary register used when data is being transferred or modified. The locate register is not saved from command to command. It is zeroed after each use. It is used as a working register by other executive functions so it must be reestablished implicitly (by the program) or explicitly (by the user) prior to its use. The edit commands are used to initially establish the register as well as to maintain the actual data referenced by these registers.

The register contents are used to control the limits of action of such statements as copy, move and rotate. There is an entry in the register for each tree level. The specified action such as exemplified above begins sequentially at the first significant (non-zero) entry, and proceeds for all data below that level. Thus, if only the cluster entry is non-zero, the specified action will take place on the entire cluster. If a component is specified, the action will apply to that component only. If a section is specified, the action will apply only to that section.

The edit commands must proceed in a hierarchical manner. A cluster must be referenced before a component, and a component must be referenced before a section.

```
BUILD      [CLUSTER
             COMPONENT] ____:____:____
             [SECTION]
```

This command provides for the creation of a new entry at the level where it is applied. If an entry by the same name already exists at this level, the older entry will be destroyed and new entry will replace it. This command can be used only for the creations of new entries (see REPLACE). The BUILD command is used to maintain the BUILD register.

```
BUILD SEGMENT ____:____:____
```

This command, although similar to the preceding commands, is not a tree structure command because it is manipulating geometry at the lowest level data structures (the segment).

The command will cause a new title to be defined in the data base in preparation for the receipt of a data block representing a sequence of X,Y,Z coordinates. SEGMENT EDIT Commands which follow will perform the actual data structuring.

```
ACCESS  [ CLUSTER
          COMPONENT ]  __:__:__
          SECTION
          SEGMENT
```

This command is used as a prelude to the manipulation of geometry within a tree structure. It establishes a sequence of pointers called the access register which identifies the geometry to be manipulated.

The ACCESS Command actually provides for the redefinition of the contents of the ACCESS Register. This register contains information which determines data to be copied. For example, the register determines the insertion position and the replacement position for INSERT and REPLACE Commands. The contents of this register also controls the data to which transformations are applied.

```
LOCATE  [ CLUSTER
          COMPONENT ]  __:__:__
          SECTION
```

This command is used as a prelude to the use of data associated with the locate name. For example, to copy or insert data from one tree structure, one would use the LOCATE Command prior to the COPY Command.

This command actually maintains the Locate Register which is essentially a temporary set of pointers for the purpose of buffering data into a tree structure controlled by the Access Register. It is used primarily by the INSERT and REPLACE Commands. If the item to be transferred using an INSERT or REPLACE Command is itself a resident of a tree, the LOCATE Commands must be used to define the data prior to the command execution.

```
INSERT  [ CLUSTER
          COMPONENT ]  ( __:__:__ )
          SECTION
```

This command will cause a new cluster component, or section to be inserted into a geometric data tree structure. The position of the insertion is defined by the Access Register and will be the position in front of the position specified by the access.

NOTE: If the data to be inserted is part of the tree, the named (__:__:__) title field must be replaced by a proper series of LOCATE Commands.

DELETE

CLUSTER
COMPONENT
SECTION

 (__:__:__)

This command will cause the specified item to be deleted from the tree. The Controlling Register is the Access Register. If the name (__:__:__) title field is omitted, the item deleted will be the item at the level specified and defined in the Access Register.

REPLACE

CLUSTER
COMPONENT
SECTION

 (__:__:__)

This command will cause one item to be replaced by another. If the new item is resident in another tree structure, the named (__:__:__) field must be preceded by an appropriate series of LOCATE commands.

COPY

CLUSTER
COMPONENT
SECTION

 (__:__:__)

This command will cause the specified data to be physically copied from its current data source into the tree structure specified by the BUILD Command. If a BUILD Command was executed prior to the COPY Command, the highest level copied will have its title changed to the title given on the BUILD Command. All other titles will remain unchanged. If BUILD Command was not given, all titles remain unchanged.

COPY SEGMENT (__:__:__)

This command can be used to copy segments into a tree structure as sections. The title of the section must be specified by a BUILD SECTION Command. Several segments can be copied into a single section by executing more than one COPY SEGMENT Command before executing a BUILD SECTION Command.

Segments may be copied into segments by executing a BUILD SEGMENT Command prior to the COPY SEGMENT Command.

The SEGMENT EDIT sub-language executes transformations which can not be performed on tree structured data. Data stored in the tree structure must be copied to segments before the Segment Edit functions can be performed. This is done by first executing a BUILD SEGMENT Command and then executing a COPY SECTION Command, and repeating for each section for which Segment Edit functions are desired.

```
ADD      [CLUSTER  
          COMPONENT  
          SECTION  
          SEGMENT]  (__:__:__)
```

This command is equivalent to a COPY Command except the data is not physically copied. Only pointers are transferred so that the proper tree linkages are established. If the (__:__:__) title field is omitted, the current Access Register is used to control the command. Titles of components and sections can not be changed by using the BUILD Command prior to an ADD Command; this would result in an error.

Output Commands:

```
LIST     [CLUSTER  
          COMPONENT  
          SECTION  
          SEGMENT]  (__:__:__)
```

This command will cause the contents of the specified tree level, or item, to be listed. Thus, LIST VEHICLE provides a list of all components in a vehicle. LIST COMPONENT provides a list of all sections in the component. LIST SECTION or LIST SEGMENT provides a listing of all points in the section or segment.

```
TREE LIST  (__:__:__)
```

This command will cause the entire tree structure of the specified vehicle to be listed.

```
LIST AVAILABLE CLUSTERS
```

This command will cause a listing of the vehicles available in the data base to be listed.

COPY BINARY (__:__:__)

COPY BCD (__:__:__)

These commands will cause the specified vehicle to be output in the IMAGE format of reference 1. The first command causes a binary file to be written. The second command causes a BCD file to be written.

If less than a full cluster is desired, the (__:__:__) field must be omitted and the output item established by the appropriate ACCESS Commands. START and STOP apply to this command.

Transformation Commands. - These commands use the right hand coordinate system with x position forward.

Rotation Parameters:

$\begin{bmatrix} \text{PSI} \\ \text{YAW} \end{bmatrix} = \underline{\hspace{1cm}}$

This is the yaw angle in degrees desired for this rotation.

$\begin{bmatrix} \text{THETA} \\ \text{PITCH} \end{bmatrix} = \underline{\hspace{1cm}}$

This is the pitch angle in degrees desired for this rotation.

$\begin{bmatrix} \text{PHI} \\ \text{ROLL} \end{bmatrix} = \underline{\hspace{1cm}}$

This is the roll angle in degrees desired for this rotation.

$\begin{bmatrix} \text{RCEN} \\ \text{ROTATION CENTER} \end{bmatrix} = \underline{\hspace{0.5cm}}, \underline{\hspace{0.5cm}}, \underline{\hspace{0.5cm}}$

These are the X,Y,Z coordinates of the center of rotation.

$\begin{bmatrix} \text{RVEC} \\ \text{ROTATION VECTOR} \end{bmatrix} = \underline{\hspace{0.5cm}}, \underline{\hspace{0.5cm}}, \underline{\hspace{0.5cm}}$

This is an alternate way of inputting the rotation angles. In this case, it is a rotation vector with I,J,K input. This does not need to be a unit vector.

[ROT
ROTATION COMMAND]

This command will cause the specified item to be rotated. The data to be transformed must have been established by an appropriate set of ACCESS Commands. The desired rotation parameters must have been established before this command is issued. The default parameters for all parameters are zero. The values established are saved and need not be changed for subsequent and identical rotations.

Scaling Parameters. - The parameter commands are:

MAG = ____

This command causes the X,Y,Z scale factors to be set to the same value.

XMAG = ____

This command sets the X scale factor.

YMAG = ____

This command sets the Y scale factor.

ZMAG = ____

This command sets the Z scale factor.

[MAGC
MAGNIFICATION CENTERS] = __'__'__

The scaling equations used are:

$$XOUT = (XIN - XC) * XMAG + XC$$

$$YOUT = (YIN - YC) * YMAG + YC$$

$$ZOUT = (ZIN - ZC) * ZMAG + ZC$$

This allows the scaling or magnification to take place about a specific point. The center of magnification (XC, YC, ZC) is input by this command.

[SC
SCALING COMMAND]

This command causes the specified scaling to be executed. The item to be scaled must have been established by a preceding set of appropriate ACCESS Commands.

NOTE: The default values for scale factors are 1.0, 1.0, 1.0 and 0.0, 0.0, 0.0 for the magnification center.

Translation Parameters.

XMOVE = ____

The X translation distance.

YMOVE = ____

The Y translation distance.

ZMOVE = ____

The Z translation distance.

[MOVE
 TRANSLATION COMMAND]

This command will cause the translation to occur. The item to be translated must have been established by a preceding set of appropriate ACCESS Commands.

NOTE: The default translation values are 0,0,0.

Bounding Commands.

START ____:____:____

STOP ____:____:____

These commands allow a Start and Stop position to be specified for a given operation. If the item being operated on is a cluster, these can be component names. If the item being operated on is a component, these can be section names.

The operation specified included the start position, the stop position and all items between.

The Start and Stop Registers are nulled after each use.

These commands can be used prior to the following commands to identify geometry to be manipulated.

COPY

LIST

COPY BINARY

COPY BCD
ALL TRANSFORMATIONS

Register Commands.

ZERO $\begin{bmatrix} \text{BUILD} \\ \text{ACCESS} \\ \text{LOCATE} \end{bmatrix}$

This command causes the specified register to be set to Zero.

$\begin{bmatrix} \text{BUILD} \\ \text{ACCESS} \\ \text{LOCATE} \end{bmatrix} = \begin{bmatrix} \text{BUILD} \\ \text{ACCESS} \\ \text{LOCATE} \end{bmatrix}$

These commands allow the contents of one register to be transferred to the other specified register.

Miscellaneous Commands.

MENU - List a menu of the available commands.

OMIT - Exit the section.

EXIT - Return to the next highest language.

Segment Edit

A segment is a sequence of X,Y,Z coordinates in three dimensional space. They are distinguished from sections in that they are not part of a data tree structure as in the case of the section. Each segment is resident in the data base under its own unique title. Therefore, any transformation can be executed on the data including transformations which increase or decrease the number of data points. The number of data points must remain the same for point level data in the geometric data tree structure.

Point Edit Commands. - The Point Edit Commands are those which apply to a single point of data. Since the definition of the internal data is the ordered set of coordinate points, each data point has 3 values and has an implied point number corresponding to the order in which it was placed in the data base. This point number is used to establish the action position for future point edit commands. The Point Edit Commands are:

* $\begin{bmatrix} P \\ \text{POINT} \end{bmatrix} = n$

$\begin{bmatrix} F \\ \text{FIND} \end{bmatrix} (X,Y,Z)$

$\begin{bmatrix} R \\ \text{REPLACE} \end{bmatrix} (X,Y,Z)$

$\begin{bmatrix} I \\ \text{INSERT} \end{bmatrix} (X,Y,Z)$

$\begin{bmatrix} D \\ \text{DELETE} \end{bmatrix} (X,Y,Z)$

$\begin{bmatrix} \text{DEF} \\ \text{DEFINE} \end{bmatrix} (X,Y,Z)$

$\begin{bmatrix} A \\ \text{ADD} \end{bmatrix} (X,Y,Z)$

$\begin{bmatrix} P \\ \text{POINT} \end{bmatrix} = n$: This command defines a point number where some subsequent action may be performed on a segment. It is referred to as the action position within the segment.

$\begin{bmatrix} F \\ \text{FIND} \end{bmatrix} (X,Y,Z)$: This command will locate the action position or point number of the point in the segment nearest the specified

Each action command has an optional title field associated with it. Each action requires two titles; the title of the segment to process and the title under which to store the processed segment.

The title of the segment to process is either the input title or the title of the last segment output, or the title of the last segment accessed, in that order.

The only way to specify an output title different from the input title is to use the BUILD Command. This command will set the output title. Immediately after execution of an action, the input title is reset to the output title, so that this title then becomes the default title.

Limit Definitions: The commands are:

START X,Y,Z

STOP X,Y,Z

NSTART = n

NSTOP = n

These commands determine the limits between which a given transformation is to take place. START and STOP use FIND to determine NSTART and NSTOP, the first and last point numbers.

Translation: The commands are:

XMOVE = ____

YMOVE = ____

ZMOVE = ____

MOVE (____:____:____)

XMOVE, YMOVE and ZMOVE set up the translation distances. The default values are zero. The command MOVE causes the actual translation to take place.

Scaling: The commands are:

XMAG = ____

YMAG = ____

ZMAG = ____

MAG = ____

$\begin{bmatrix} \text{MAGC} \\ \text{MANIFICATION CENTER} \end{bmatrix} = X_C, Y_C, Z_C$

$\begin{bmatrix} \text{SC} \\ \text{SCALE} \end{bmatrix} (_, _, _)$

XMAG, YMAG and ZMAG are the magnification factors applied during the scaling operations. MAG sets all of the magnification factors to the same value. The default magnification values are 1.0. The transformation equations are:

$$X_T = (X_I - X_C) * XMAG + X_C$$

$$Y_T = (Y_I - Y_C) * YMAG + Y_C$$

$$Z_T = (Z_I - Z_C) * ZMAG + Z_C$$

The Command MAGNIFICATION CENTER = X_C, Y_C, Z_C establishes the values as X_C, Y_C, Z_C . The default values are zero. The Command SCALE causes transformation to be executed.

Rotation: The ROTATIONS Commands are:

$\begin{bmatrix} \text{PSI} \\ \text{YAW} \end{bmatrix} = _ \text{ (Degrees)}$

$\begin{bmatrix} \text{THETA} \\ \text{PITCH} \end{bmatrix} = _ \text{ (Degrees)}$

$\begin{bmatrix} \text{PHI} \\ \text{ROLL} \end{bmatrix} = _ \text{ (Degrees)}$

$\begin{bmatrix} \text{RCEN} \\ \text{ROTATION CENTER} \end{bmatrix} = \underline{X_C}, \underline{Y_C}, \underline{Z_C}$

$\begin{bmatrix} \text{ROT} \\ \text{ROTATE} \end{bmatrix} (_ : _ : _)$

YAW, PITCH and ROLL establish the rotation angles of the transformation. The default values are zero. ROTATION CENTER

establishes the center of rotation of the transformation. The Command ROTATE causes the transformation to be executed.

Cutting a Segment with a Plane:

Plane Definition: The Plane may be defined and used in the rotation command. In this case RCEN is a point on the Plane, and PSI, THETA and PHI are angles describing the direction of the normal.

PLANE = $X_C, Y_C, Z_C, \text{PSI}, \text{THETA}, \text{PHI}$: This is a one-line command setting all the values described above.

XCUT = ____: This assumes a YZ Plane passing through the specified X with a direction of positive X.

YCUT = ____: This assumes a XZ Plane passing through the given Y value in the direction of positive Y.

ZCUT = ____: This assumes a XY Plane passing through the given Z with a direction of positive Z.

PCUT (____:____:____): This command will cause all of the points in the direction of the positive normal, plus all plane intersections to be output.

MCUT (____:____:____): This command will cause all of the points in the direction of the negative normal, plus the plane intersections to be output.

Point Redistribution: The commands are:

NSEG = ____

EQLEN (____:____:____)

These commands will cause the segment to be redistributed such that the arc length of the line described by this space is divided into NSEG equal positions. This means that NSEG + 1 points are output to describe this point redefinition.

Data Acquisition: Four CLUSTER EDIT Commands are included to allow the user access to tree stored section data. These commands are:

*ACCESS CLUSTER ____:____:____

*ACCESS COMPONENT ____:____:____

point such that $(X_S - X_F)^2 + (Y_S - Y_F)^2 + (Z_S - Z_F)^2$ is a minimum.

In some cases there will be multiple points of the same value. For example, a cross section closing on itself will have identical first and last points. This situation is handled by the use of additional calls to FIND. If FIND already has been called, then the search for the point begins at the next point after the one found by the previous command.

$\left[\begin{array}{c} R \\ \text{REPLACE} \end{array} \right] (X,Y,Z)$: This command will cause the point specified by action position to be replaced by the given X,Y,Z values. If omitted, the point specified by the DEFINE Command will be used.

$\left[\begin{array}{c} I \\ \text{INSERT} \end{array} \right] (X,Y,Z)$: This command will cause the given X,Y,Z values to be inserted in front of the action position. If X,Y,Z is omitted, the point specified by the DEFINE Command will be used.

$\left[\begin{array}{c} D \\ \text{DELETE} \end{array} \right] (X,Y,Z)$: This command will cause the X,Y,Z input point to be deleted, and if omitted, the point specified by the action position will be deleted.

NOTE: If the X,Y,Z is input, a procedure similar to FIND is used to determine the point to be deleted.

$\left[\begin{array}{c} \text{DEF} \\ \text{DEFINE} \end{array} \right] (X,Y,Z)$: This command can be used in place of the X,Y,Z inputs in all of the point commands except POINT and DELETE.

$\left[\begin{array}{c} A \\ \text{ADD} \end{array} \right] (X,Y,Z)$: This command will cause the input X,Y,Z values to be added to the end of a segment. If X,Y,Z is omitted, the point defined by the DEFINE Command will be used.

NOTE: The values determined by POINT and FIND are zeroed after each use. They are not maintained.

Segment Level Commands. - Segment level commands are those which apply or transform to an entire segment. The commands fall into two groups - informational and action.

*ACCESS SECTION __:__:__

*COPY SECTION (__:__:__)

These commands will allow the user to construct segments from existing sections. See the CLUSTER EDIT Commands for a complete description.

ACCESS (__:__:__): This command will access a stored segment. The title of this segment is established as the input or active title.

Segment Creation:

BUILD (__:__:__): This command will cause a new segment to be built. It establishes the output title for any transformation for the ADD X,Y,Z Command the the COPY Command.

COPY (__:__:__): This command will cause the data stored under the specified title to be copied to the title specified by the BUILD Command.

Data Display:

[L]
LIST (__:__:__): This command will cause a listing of the specified segment to be printed.

Miscellaneous Commands:

MENU

OMIT

EXIT

These are the general utility commands and have the same meaning as described in CLUSTER EDIT.

Program Output. - Since the GTM program is basically a geometry manipulation tool and highly interactive, program output essentially remains transparent to the user. They are, however, two types of output which are applicable. The first is those which can be classified as geometric analysis outputs. The second is formatted geometry.

Geometric analysis output includes mass properties evaluations, program response to user input, geometric displays and geometric parameters. The formatted geometry output is the cornerpoint geometry sets used by other technology modules.

HABACP: HYPERSONIC ARBITRARY BODY AERO-
DYNAMIC COMPUTER PROGRAM.

The input to HABACP has been slightly modified to include a namelist input to replace the former option card (card 1). The new input is \$HYPIN and has one input IPG which the array of option numbers desired by the user. A sample input as as follows:

\$HYPIN IPG=1,1, \$

The EDIN version of the Hypersonic Arbitrary Body Aerodynamics Computer Program has been modified to provide complete storage of aerodynamic coefficients in the dynamic data base. The coefficient arrays are stored in namelist format on the special EDIN output file by name. The stored array lengths are determined by the input quantity, NAB (Card Type 9, Column 61-62). Each vehicle section may have a different NAB. The names are generated from the root names, examples in figure 14. The root names are correlated with the force data headings which appear on the printed output (i.e. CL = CL, CD = CD, etc.). The three-digit integer identification, CASE*, is concatenated to the root name to form the complete EDIN name. CASE is printed in the upper left corner of each tabulated page of printout. If no value for CASE is input, then only the root name will be used.

Using the above described technique, the user may modify the construction of the names generated in the HABACP by the CASE input. For example, suppose the vehicle consisted of three sections, body, wing and vertical tail. Suppose also that the user wished to store selected summations for the body alone, the wing alone, the wing-body and the complete configuration. The CASE input might be as follows:

<u>COMPONENT</u>	<u>CASE INPUT</u>
BODY	010
WING	020
WING-BODY SUMMATION	021 (CASE INPUT ON SUMMATION CARD TYPE 1)
VERTICAL TAIL	030
WING-BODY=VERTICAL SUMMATION	031 (CASE INPUT ON SUMMATION CARD TYPE 1)

*(Card Type 8, Column 66-68 for component.
Card Type 1, Column 66-68 for summation.)

<u>FORCE DATA HEADING</u>	<u>ROOT NAME</u>	<u>DESCRIPTION</u>
ALPHA	ALP	Angle of Attack (degrees)
BETA	BET	Angle of Yaw (degrees)
CD	CD	Drag Coefficient
L/D	LOD	Lift Drag Ratio
CL	CL	Lift Coefficient
CM	CLM	Pitching Moment Coefficient
CA	CA	Axial Force Coefficient
CY	CY	Side Force Coefficient
CN	CN	Normal Force Coefficient
CLL	CLL	Rolling Moment Coefficient
CLN	CLN	Yawing Moment Coefficient
CF	CF	Skin Friction Coefficient

Figure 14 ROOT-NAME DEFINITIONS FOR ODIN/HABACP INTERFACE

The stored data would consist of all the force data arrays listed in figure 14 for each of the above combinations of vehicle components. The generated names for the above example would be as illustrated below:

Angle of Attack Array -

ALP10, ALP20, ALP21, ALP30, ALP31

Lift Coefficient -

CL10, CL20, CL21, CL30, CL31

Notice the trailing zeros are part of the generated name; leading zeros are not.

The new EDIN/HABACP interface required modification to the subroutine AERO. The modification required a number of calls to a new subroutine ADDREL. The calls are located immediately following the normal printout statements. Subroutine ADDREL and the associated function NAMGEN are part of the EDIN Data Management System. The two routines generate the name oriented output data described above.

IMAGE DISPLAY COMPUTER PROGRAM

The IMAGE graphics software package provides the capabilities to access and display cornerpoint geometry. The program utilizes a virtual graphics display technique which allows the user to display multiple views, zoomed views of a specific region or component and a windowing feature in which the user can specify the display region on the viewable screen. In addition, a symbolic routine is provided for picture labeling.

The Display Device

The IMAGE computer code is a software package designed to display in pictorial form figures built up from combinations of cornerpoint geometry. The primary display device is the Tektronix 4012 display tube, however, the plotting routines also interface with the CALCOMP plotter for hardcopy drum plots, the SD4060 for microfilm plots and the high speed online MOPS Hazeltine 4000.

The software provides the user with the ability to display multiple views, zoomed views of selected parts, windowing and picture labeling. The basic description of IMAGE, its input requirements, operational characteristics and output are described in reference 27. The basic extension to IMAGE was the installation of the virtual graphics display capability.

One may consider a set of points in 3-D space, the limits of which dictated by the numeric capability of the host computer. Upon this "virtual" space is then imposed a window through which orthographic projections of the object are observed. This window is plotted to user specified scale.

Virtual Graphics

Operations which are applied to the screen are called direct graphics while those which are applied to the virtual display are known as virtual graphics. The virtual display is a two-dimensional surface of undeterminable size, limited only by the numeric processing of the computer. The user is only responsible for defining what portion of the view he wishes to be displayed, e.g. a zoomed view. The virtual display is similar to normal plotting devices in that there is a movable point which may be thought of as a writing cursor on the virtual display.

All or any portion of the virtual display may be viewed at any time through the technique of windowing. The portion of the virtual display to be shown is defined by rectangular boundaries. This boundary is called the virtual window and only those vectors which pass through the window are displayed.

It is not necessary to use all of the screen for display of the virtual window. The user may define a rectangular section at any location on the viewable screen for display of the virtual image. Figure 15 illustrates the technique of windowing.

Program Usage

The program usage requirements described in the following paragraphs are orientated towards the Tektronix/Exec 8 installations. The program input requirements and available options correspond to the IMAGE virtual graphics package only.

The use of the package requires two types of inputs, the control instructions and the actual program inputs. Figure 16 illustrates the instruction setup for executing the module.

General Program Input

The IMAGE program input consists of three types:

1. Program controls (\$TYPE32) in namelist format.
2. Surface model data as formatted cornerpoint geometry.
3. Picture specifications (\$TYP343) in namelist format.

The program controls and picture specifications are input in standard namelist input format. The namelist reads are provided for this purpose.

When a namelist read is encountered in the program, the entire input file is scanned up to an end-of-file or a record with a dollar (\$) in column 2 followed by the namelist name requested by the IMAGE program. Succeeding data items are read until a second dollar (\$) is encountered signifying the end of the namelist. All data between the opening and closing dollar is interpreted by the namelist input routine.

\$TYPE32 Inputs

The first list (\$TYPE32) is used for specifying general data pertaining to program control including geometry file control. The following paragraphs define each \$TYPE32 namelist input in detail. Each paragraph is headed by the name and default value for the variable listed.

Usually the configuration geometry consists of more than one component. In the data format of the IMAGE program, the data for each component is terminated with an integer flag referred

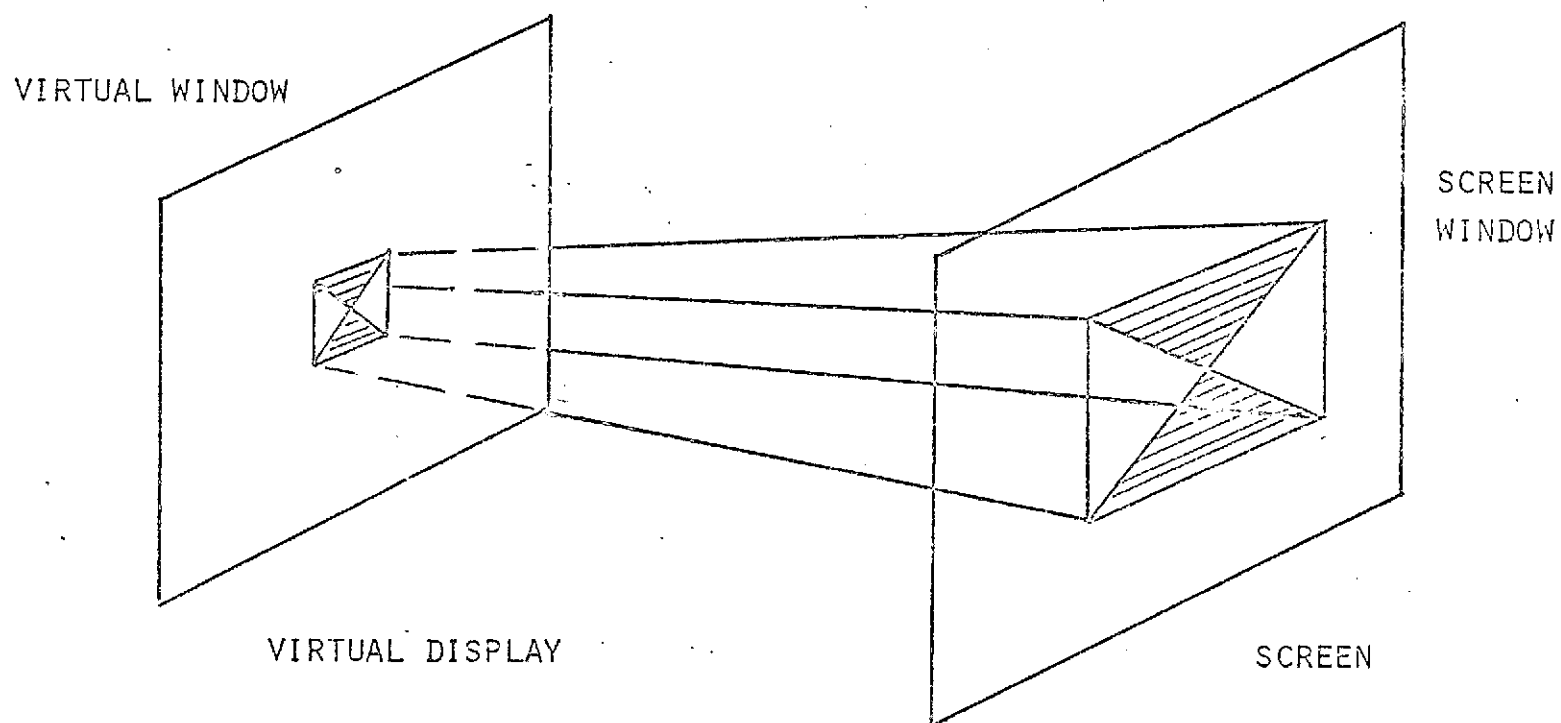


FIGURE 15 ILLUSTRATION OF THE IMAGE WINDOWING TECHNIQUE.

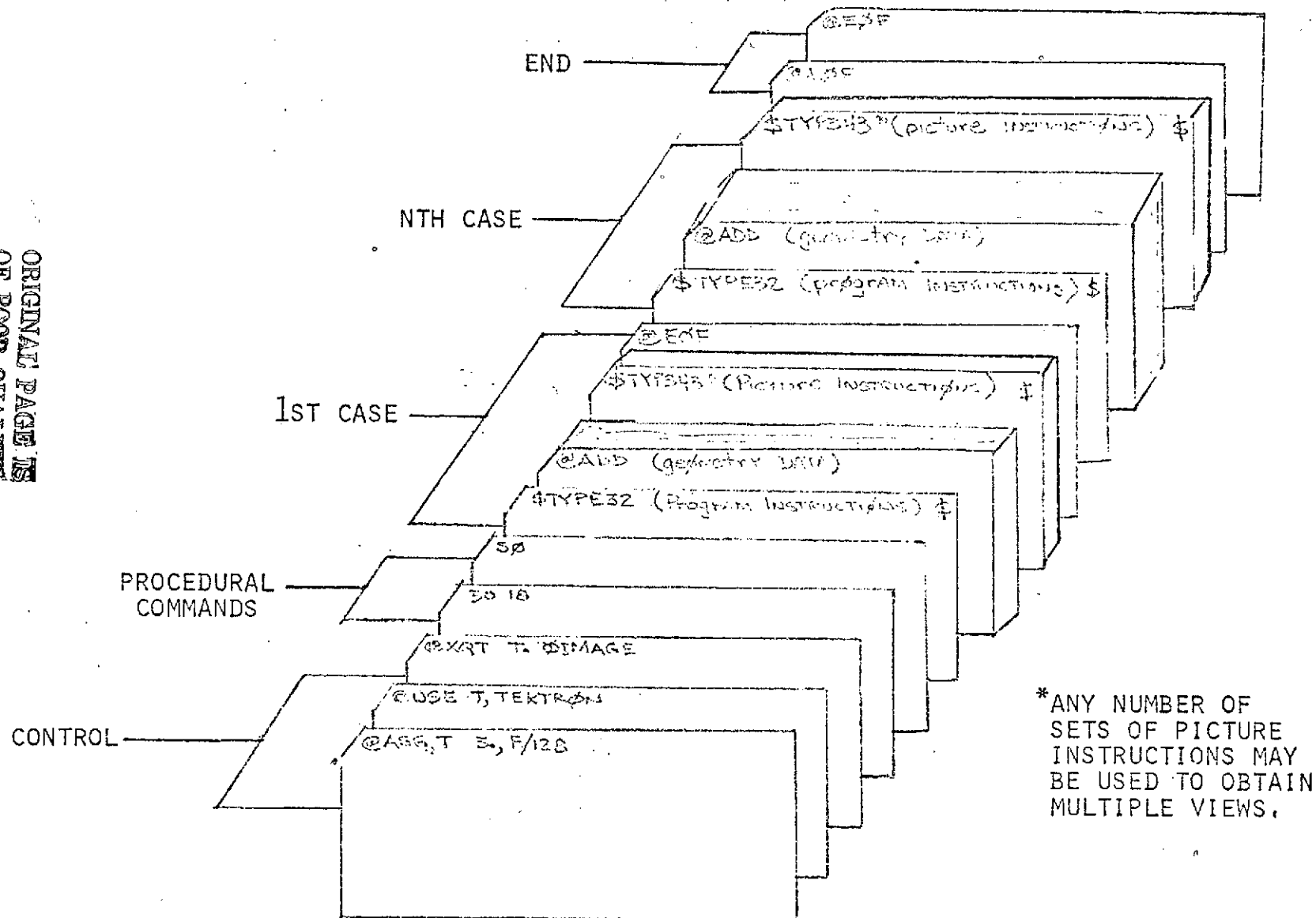


FIGURE 16 IMAGE RUN STREAM

to as a status flag as described under Surface Model Data (below). The merging of the geometric components is desirable for plotting purposes. Therefore, the geometric components are merged into a single component as the geometric data is being transferred to the temporary file, TAPE3. The user of the program must specify the number of components to be merged for each picture sequence. This specification is accomplished by the integer input variable ISTAT3.

ISTAT3 = 1 Integer variable number of vehicle components with Status = 3 in the vehicle geometry which the user wishes to plot as a unit. The program will count the number of Status = 3 in the geometry deck and when the count reaches this input value, the program will proceed to the plot options for the vehicle components which have just been read.

Several options are available to the user of the program for accessing geometric data. Alternate files may be employed and alternate formats may be specified. The input parameter for controlling these options is the integer variable ITAPE.

ITAPE = 0 Geometry tape control integer variables with the following possible values:

- = -1 Geometry data (type 3) already exists on unit 3 in a suitable form for display. No data will be read.
- = 0 Geometry data (type 3) will be read from Tape 5 (geometry data cards are loaded along with picture-data control cards).
- = 1 Geometry data (type 3) will be read from the geometry storage tape (tape 8) in coded format.
- = 2 Geometry data (type 3) will be read from the geometry storage tape (tape 8) in binary format.

The IMAGE program provides a flexible means of controlling the alternate geometry file, TAPE8. Multiple components may be stored on TAPE8. These components may be extracted in sequential groups or plotted individually. The input parameter for rewinding the geometry tape (TAPE8) is IREW8. Usually, the

file is rewound for the first sequence of pictures, then through the use of the input variable ISTAT3, additional groups of components can be extracted for plotting purposes.

IREW8 = 0 Integer variable to control the position of Tape 8 just before the geometry data is read from it.

 = 0 Rewind Tape 8 and then read geometry data from it.

 = 1 Do not rewind Tape 8, but start reading geometry data from it in its current position.

The integer, IREW8, permits the user to store more than one group of vehicle geometrics on the geometry tape, then collect and plot them by groups.

Geometry Data

The standard method for inclusion of geometry data in the run stream is through the use of an @ADD control card. Geometry data must be correctly formatted in accordance with reference 27.

To aid the user, a number of file elements containing formatted geometry data have been created. These elements reside on the file EX42-00002.DATA8.

\$TYP343 Inputs

After the geometry has been added, the second namelist (\$TYP343) is input. This data set provides input for picture control options e.g. zooming, view orientation, labeling, windowing and scaling. Figure 17 illustrates the input logical flow for IMAGE. Any number of views may be specified by inputting only one @EOF after a TYP343 input. An exit from IMAGE is performed when an input of @EOF is followed by another input of @EOF.

The following defines the TYP343 input variables and their default values:

DXG	0	DXG defines the X axis origin (in inches) of the screen window. Figure 18 illustrates the orientation
-----	---	---

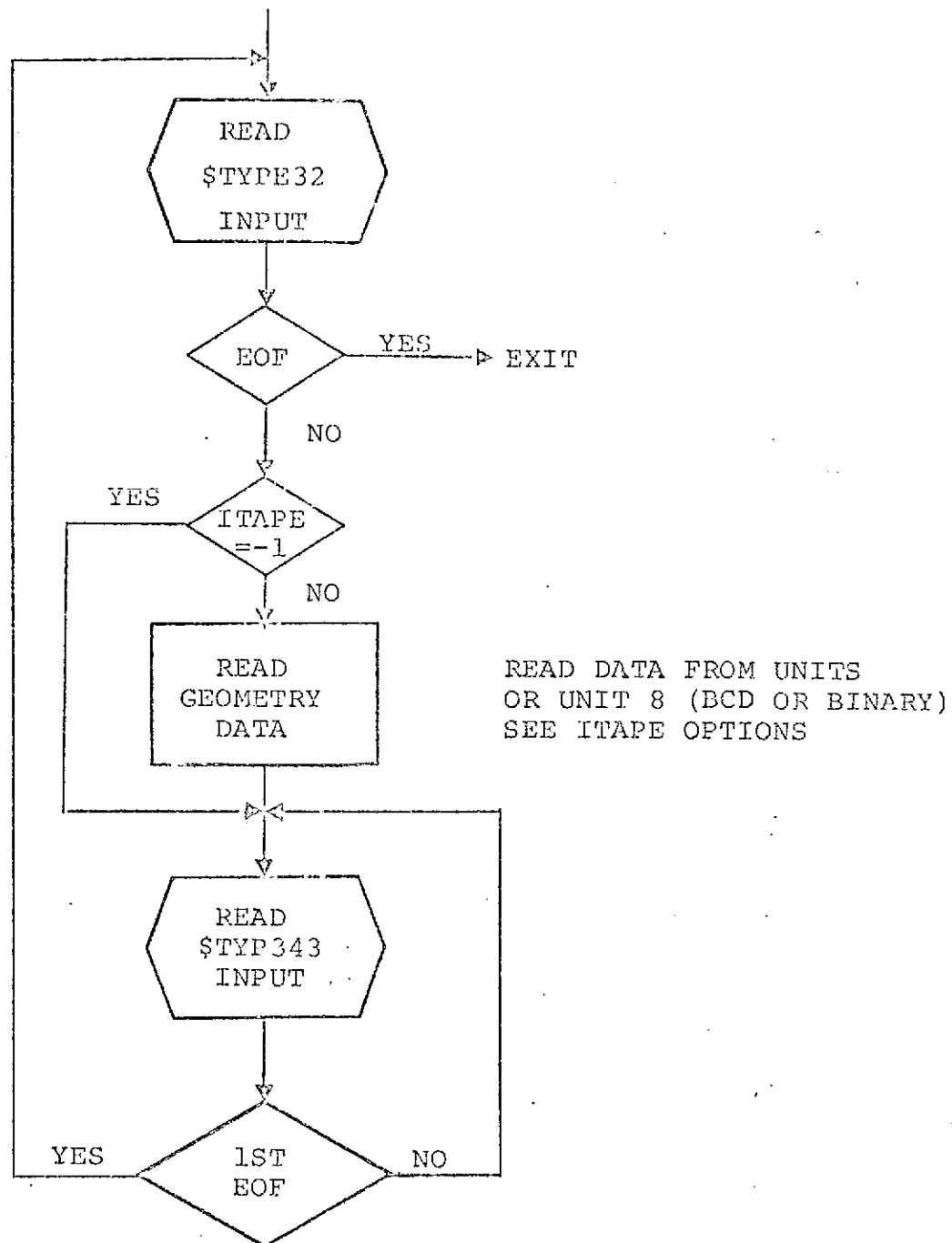


FIGURE 17 ILLUSTRATION OF INPUT FLOW LOGIC FOR IMAGE.

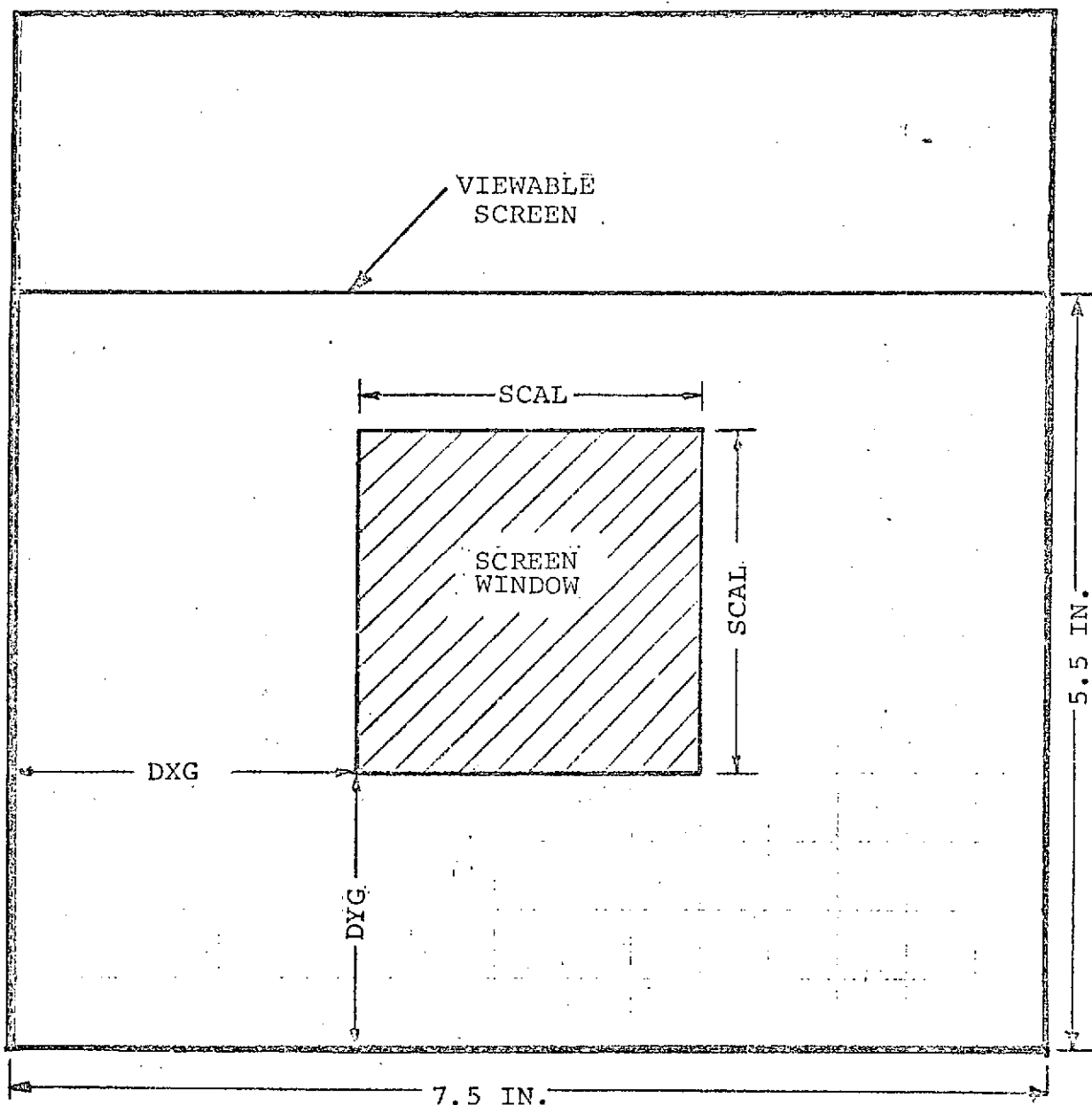


FIGURE 18 ORIENTATION OF THE SCREEN WINDOW.

of DXG with the screen window.

NOTE: When DXG is used in conjunction with the TEXT option, it defines the first character position.

DYG	0	DYG defines the Y axis origin (in inches) of the screen window. Figure 18 illustrates the orientation of DYG with the screen window. NOTE: When DYG is used in conjunction with the TEXT option, it defines the first character position.
SCAL	7.5	Size of the screen window (in inches) in both the X and Y directions. Reference figure 18.
PHI	0.0	Angular orientation (in degrees) of the viewing plane with respect to the X axis. See figure 19.
THETA	0.0	Angular orientation (in degrees) of the viewing plane with respect to the Y axis. See figure 19.
PSI	0.0	Angular orientation (in degrees) of the viewing plane with respect to the Z axis. See figure 19.
SCALV	(Initially Computed)	Size of the virtual window in both the X and Y directions. This parameter is primarily used for zooming.
DELX	(Initially Computed)	X axis translation of the geometry coordinates to the display axes. Used in conjunction with view translations and zooming.
DELY	(Initially Computed)	Y axis translation of the geometry coordinates to the display axes. Used in conjunction with view translations and zooming.

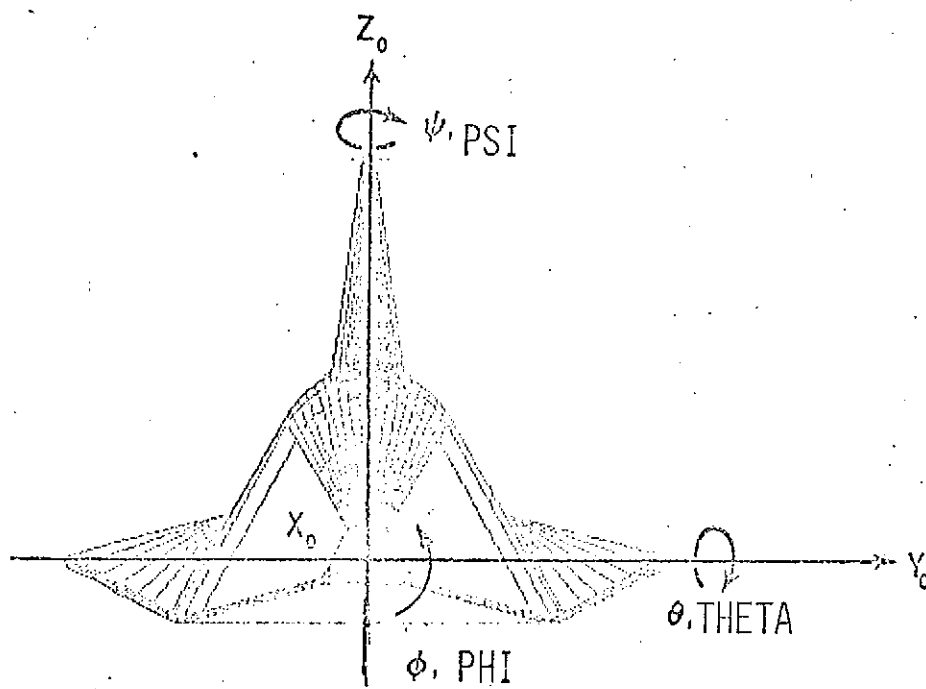


FIGURE 19 ANGLE CONVENTION FOR THE VIEWING PLANE.

ORIGINAL PAGE IS
OF POOR QUALITY

DELZ	(Initially Computed)	Z axis translation of the geometry coordinates to the display axes. Used in conjunction with view translations and zooming.
WINDO	F	Logical variable for drawing a border around the defined screen window.
TEXT	F	Logical variable for text input. = T Text information will follow TYP343 input. = F No text information will be input.
HTEXT	.07	Height of the characters (in virtual coordinates) in the text material.
ISHAD	F	Hidden line option. T = Plot hidden lines. F = Delete hidden lines.
IREFL	1	Reflected element flag. Provides the option of plotting only the input geometry or plotting the input geometry and the reflected geometry. 1 = Draw reflected geometry. 0 = Draw input geometry only.
COPY	0	Logical variable, if true a hard copy will be generated after the current view is generated.
PAGE	0	Logical variable, if true the viewable screen will be erased before the current view is generated.
IQUAD	0	Corner point plot flag. 0 = Plot actual corner points. 1 = Plot quadrilateral corner points.

PANEL: GEOMETRY GENERATION PROGRAM

The PANEL Program is a collection of general purpose geometry definition subprograms developed for use in preliminary configuration analysis. The PANEL Program consists of geometry subroutines from reference 31 and other geometric generation programs.

The program produces a vehicle surface definition in the form of a sequence of quadrilateral panels defined by their four corner points. The resulting cornerpoint data is acceptable as input to the Hypersonic Aerodynamic Program of reference 31, GTM, IMAGE or other technology programs in the EDIN system. The input data for the assembled program is namelist and the order depends on the subprogram options.

Program Description

The panel program has been structured into a sequence of first level overlay elements that generate three dimensional geometric shapes for use by the EDIN programs. The elements of the programs are linked through the use of the Univac Exec 8 MAP processor. The subprogram options and input flow logic are illustrated in figure 20.

Logical selection to the geometry generators in the subprograms is controlled by \$IPANEL. After completion of any generations, model control is returned to the PANEL main program.

The coordinate system used in the program is the right hand Cartesian system. In the convention used, vehicle is usually positioned with the x-axis point forward, the y-axis pointing to the right, and the z-axis point upward. The wing and body surfaces are represented by sets of points in space called panels. Each pair is a set of four points connected by straight lines.

COPY5 Module. - The COPY5 module allows the user to input the corner points of geometric panels.

The points on the body surface are input in rows and columns. The number of panels in the whole section is defined by the number of rows of panels times the number of panels per row. The orientation of the geometric section is optional but two rules must be followed regardless of the orientation.

1. Points along a row are input sequentially upward.
2. Rows of points are input sequentially to the right.

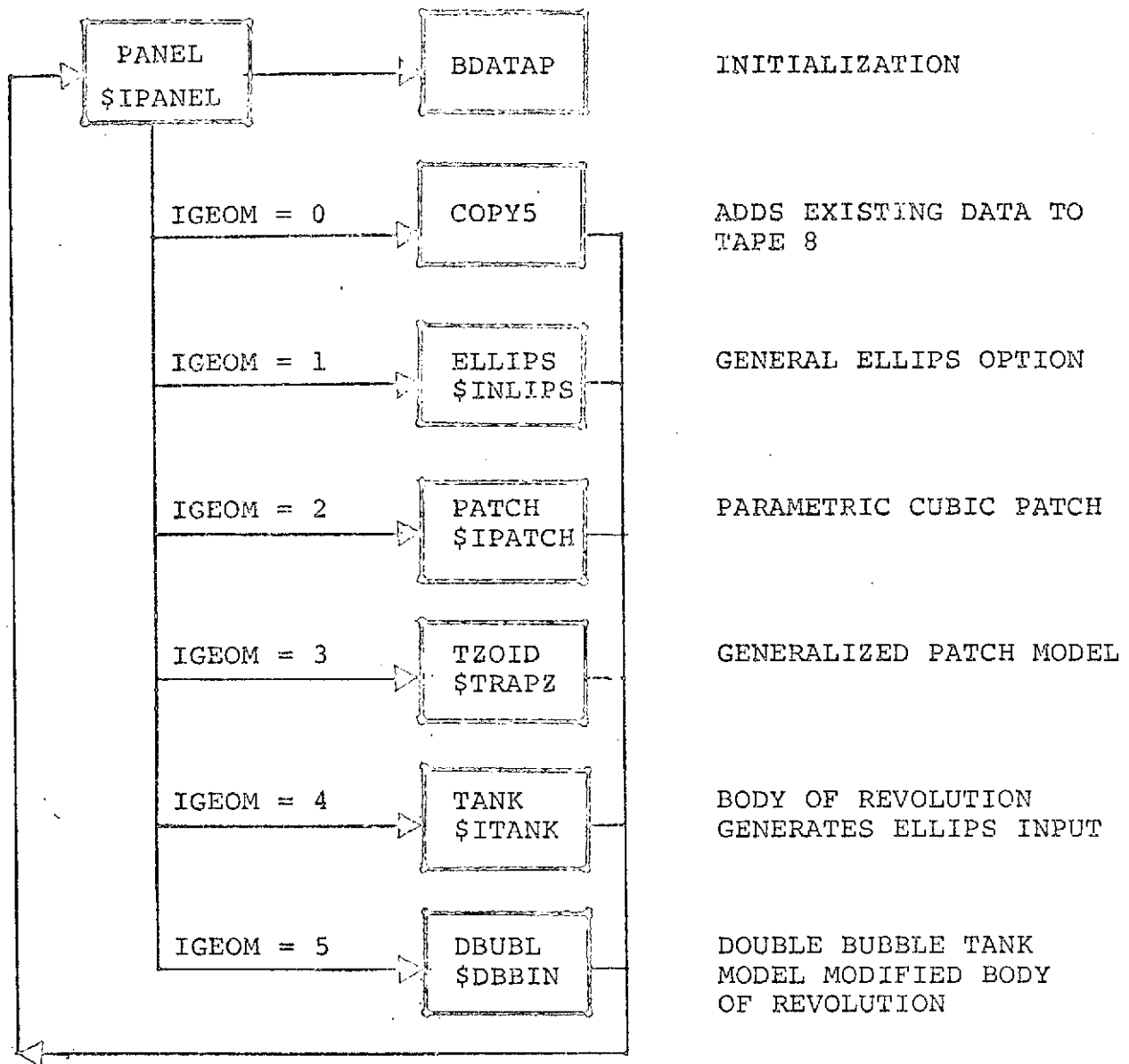


FIGURE 20 GEOMETRY GENERATOR OPTIONS.

ELLIPS Module. - The ELLIPS module allows the user to generate panel information for partially or completely elliptical cross sections. The surface of the section is described by an ellipse centered at some point off the reference axis and defined by the major and minor axis as shown in figure 21. The portion of the reference ellipse used to define the body section is defined by the angular difference between θ_0 and θ_L measured from the negative z axis. A sequence of two or more sections describe a surface. The PANEL Program generates the panel geometry for an arbitrary number of sections. Each section can be equally divided into an arbitrary number of divisions.

Cubic Patch Module. - The cubic patch is provided as an alternate input method in the description of arbitrary shapes. In this respect, it serves the same purpose as the surface-element input method.

In the panel cornerpoint input method a vehicle section is described by a large number of surface points organized in panel fashion. In the cubic patch method only points along the boundaries of a patch are input to the program, and the distributed surface points required for the subsequent panel calculations are determined by the program.

The basic features of this method are that fewer input points are required to describe a surface and the generated panel size is controlled by two input parameters and may be changed to meet the requirements of the problem.

Figure 22 illustrates how a section is described by this method. Each of the four boundaries is identified in this figure; two in the w direction and two in the u direction. The input data for each of these boundaries can be input by array name. Boundary 1 is defined by three arrays, XW1, YW1, ZW1, representing the coordinates with respect to the reference system. Boundary 2 is defined by the three arrays, XW2, YW2 and ZW2. Similarly, boundaries 3 and 4 are defined as XU3, YU3, ZU3 and XU4, YU4 and ZU4, respectively.

The user orientates the model of the vehicle so that the number 1 boundary is to the left and the number 2 boundary to the right. The order of the points is from the bottom to the top of the patch. Note that a point must be included outside the patch at either end of the boundary to give proper slopes at the cornerpoints. Boundaries 3 and 4 are loaded from left to right. A different number of points may be used to describe each boundary up to a maximum of 20 for each.

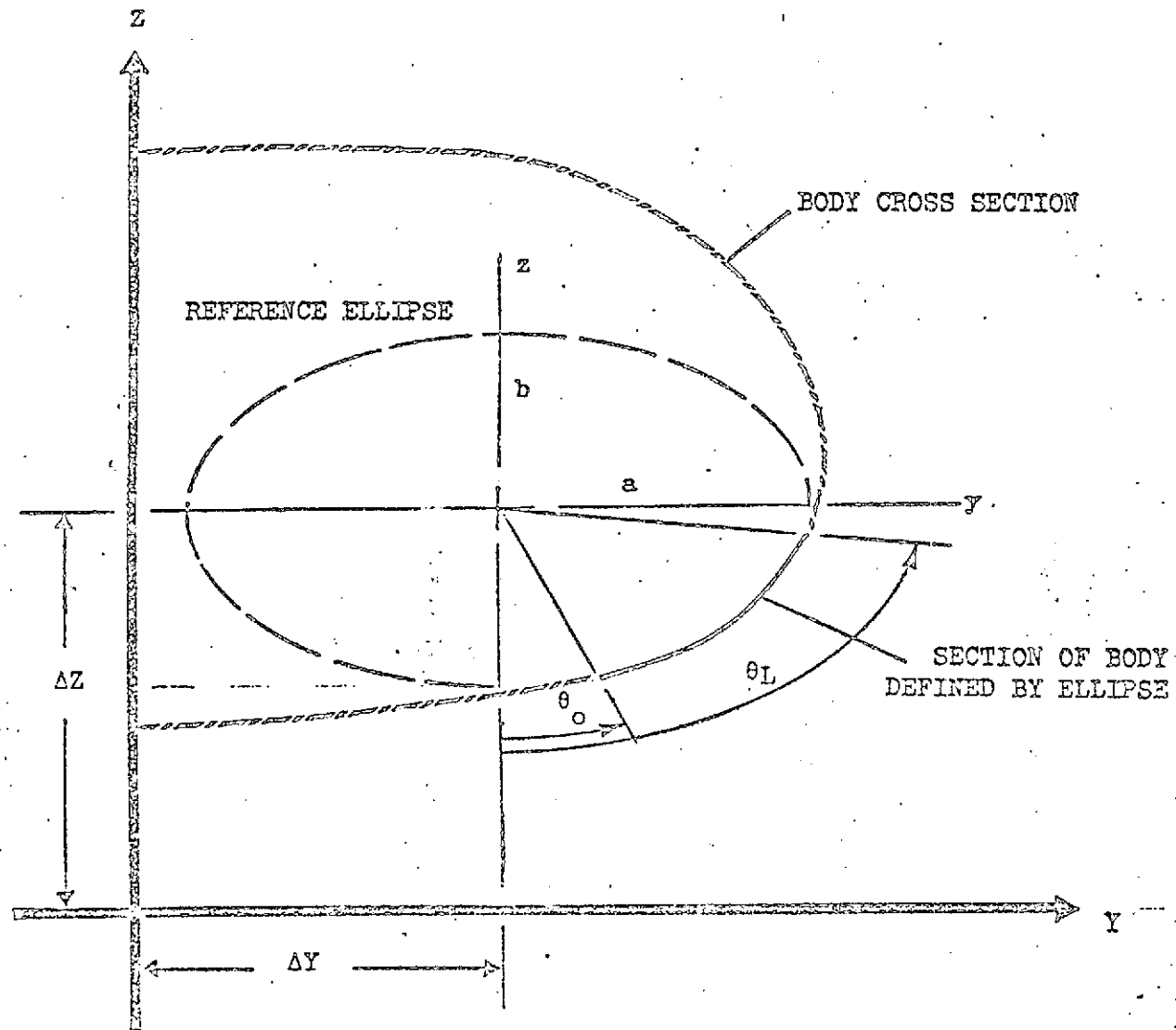


FIGURE 21 GEOMETRY DEFINITION FOR ELLIPTIC METHOD.

ORIGINAL PAGE IS
OF POOR QUALITY

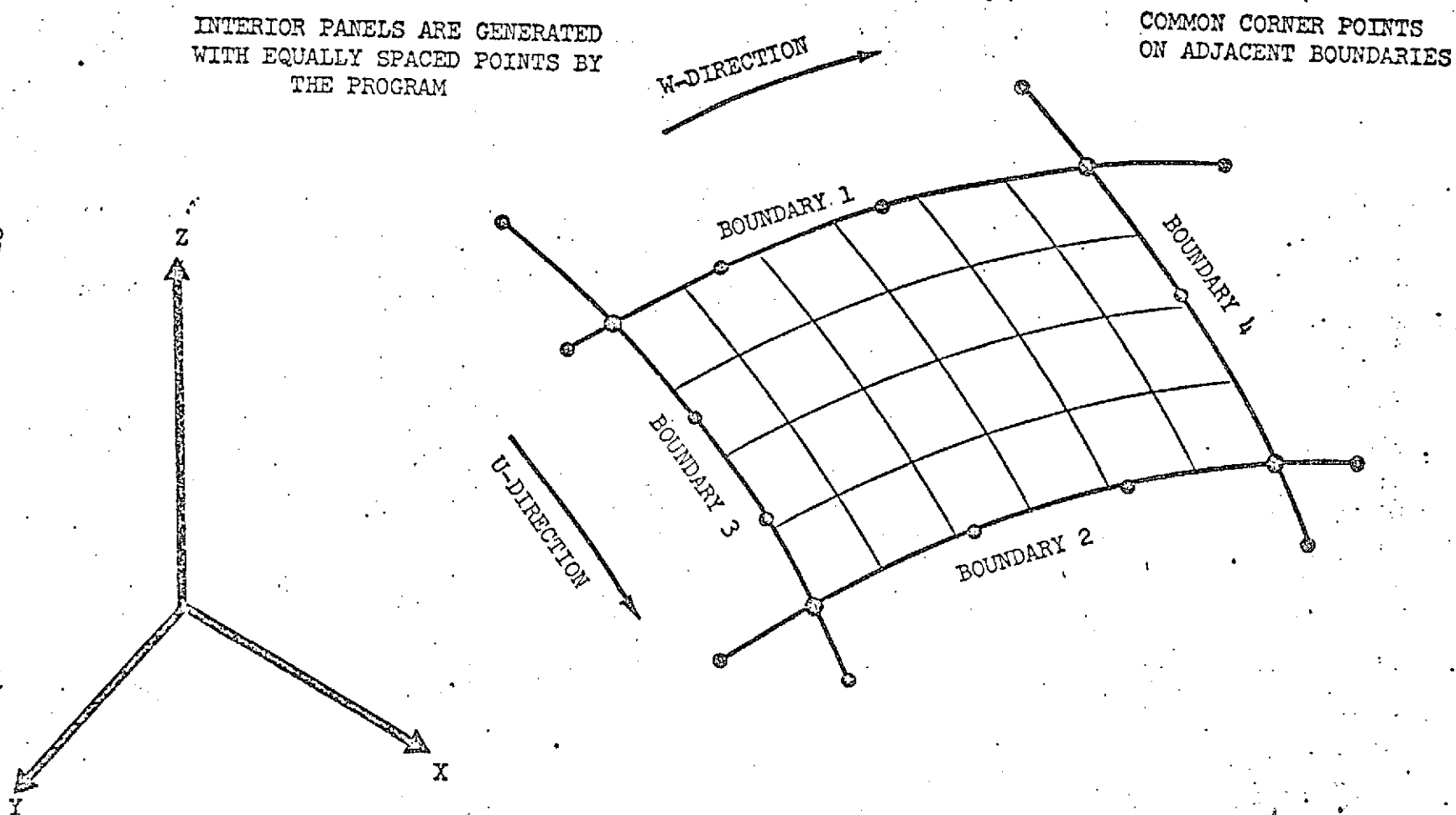


FIGURE 22 INPUT DESCRIPTION FOR THE CUBIC PATCH TECHNIQUE.

Each boundary curve must be extended by one point on each end to permit the computation of end point derivations. The second point and the next to the last point on each curve must be common to the adjacent curves, as illustrated in figure 22. The program generates equally space panels based on arbitrary numbers of rows and columns of panels selected by the user.

TANK Module. - The TANK model generates the x-coordinates along a center line and the corresponding radii based upon simplified geometry descriptions. The x-coordinates and radii can then be communicated to the ellipse model (already available in PANEL) for the actual cornerpoint geometry generation. Figure 23 illustrates some of the options available in the tank and engine model. In addition to those shown, third order nose transitions can be generated by specifying the initial ramp angle at the point of tangency to the nose radii. Also, arbitrary base radii may be specified providing the capability of generating engine nozzles.

Double Bubble Tanks. - The DBUBL tank model, figure 24 , calculates the parameters required to generate double bubble tanks in the ELLIPS model. The user specifies the position, rotation, radius and separation distance. The subprogram calculates the initial and final angles necessary to generate the bodies of revolution representing the tanks.

TZOID Module. A TZOID Model body shape generator has been developed for the PANEL program based upon the trapezoid. The method uses straight line segments joined by radii. The shape is completely controlled by the position and magnitude of the radii. Therefore, the method is not restricted to trapezoidal shapes but can degenerate into circles, rectangles, triangles or flat plots as illustrated in figure 25.

Physical Characteristics

The physical characteristics of the panel program are summarized as follows:

HOST COMPUTER:	UNIVAC EXEC 8 (1110)
PROGRAM FILE(S):	EX42-00002*PANEL. (SOURCE, RELOCATABLES ABSOLUTE)
	EX42-00002*PANEL. (MAP ELEMENT)
ABSOLUTE ELEMENT:	OPANEL
LANGUAGE:	FORTRAN V
PROGRAM SIZE:	19850 DECIMAL
CARD SOURCE:	4000

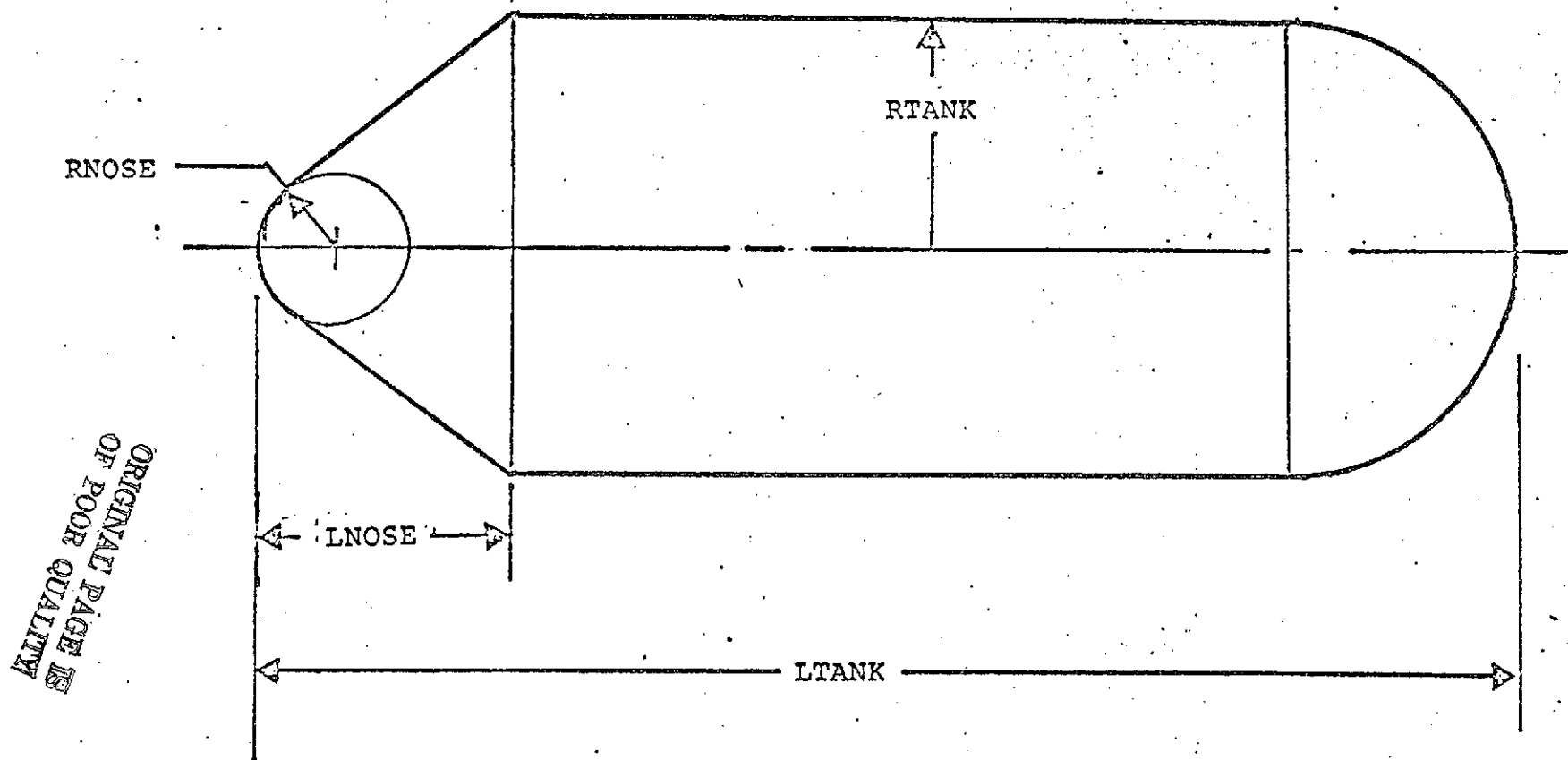


FIGURE 23 TANK AND ENGINE MODEL.

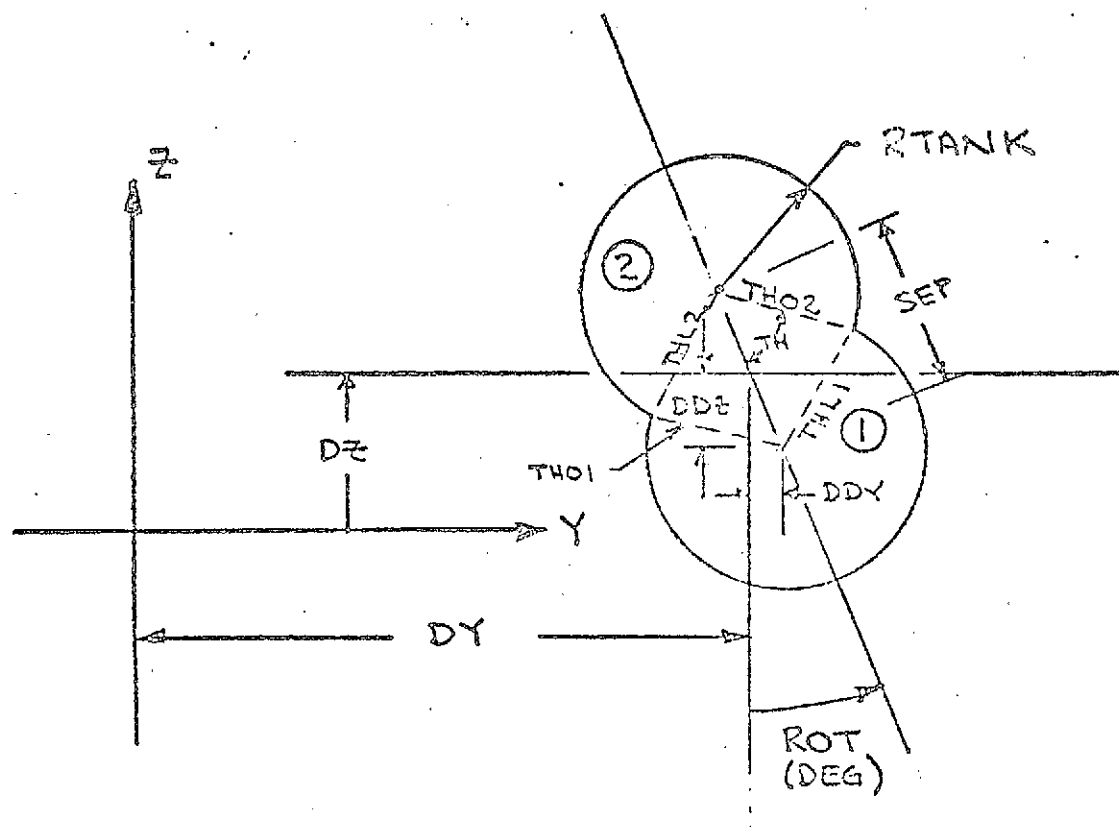
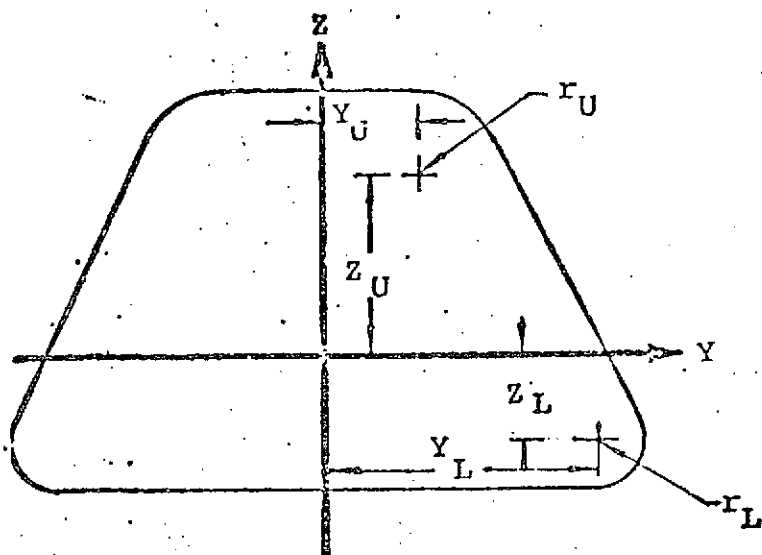


FIGURE 24 DBUBL TANK MODEL GEOMETRY.



SYMBOLS

r = radius

Y = Y coordinate of center of radius

Z = Z coordinate of center of radius

SUBSCRIPTS

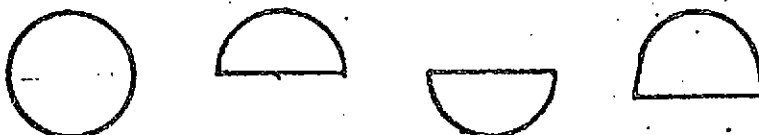
U = upper

L = lower

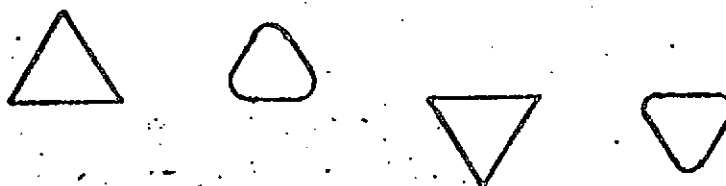
POSSIBLE BODY SHAPES



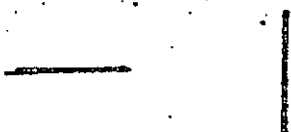
RECTANGULAR



CIRCULAR



TRIANGULAR



FLAT PLATE

FIGURE 25 GENERALIZED TRAPEZOIDAL SECTION.

Program Usage

The computer program usage requirements described in this section are generally orientated toward the Exec 8 1110 version and specifically towards the Johnson Spacecraft Center installation. The actual program input requirements described are applicable wherever the program is installed but the control cards for the retrieval and execution of the program will differ from computer to computer. Figure 26 illustrates a typical run stream for PANEL.

There are five namelist input groups. The namelist IPANEL is read before each section is input and describes the type of geometry and the input format to be used. It contains the following data:

```
IGEOM    = 0, cornerpoint geometry will be input for this
           section.
           = 1, elliptic cross-sectional data will be input
           for this section.
           = 2, cubic patch data will be input for this
           section.
INFORM    = 1, HABACP input format of reference 27 will be
           used in this section.
           = 2, namelist input format will be used for this
           section.
IPRINTS   = 0, no printout of panel characteristics.
           = 1, print panel characteristics.
ISTOP     = 0, more geometry data is coming.
           = 1, stop the program.
```

The namelist \$INLIPS may be used to describe elliptical input in the following manner: Set IGEOM = 1 and INFORM = 2 in the namelist \$IPANEL.

```
NPTS      - number of sections used to describe the surface.
AX(I)      - array of longitudinal positions of the sections.
DELZX(I)-  array of ellipse offsets in the z direction for
           the sections.
DELYX(I)-  array of ellipse offsets in the y direction for
           the sections.
THETOX(I)  array of angles, in degrees, which defines the
           first points on the ellipses used to describe
           the body sections.
```

	<u>ID</u>	<u>ACCOUNT NUMBER</u>	<u>ORGANIZATION</u>
@RUN	AB000,	AB111 - Z111 - L,	EX42-00002
@ASG,T	8.,F///500		{ Assign temporary file 8 for geometry output file.
@XQT	EX42-00002*	PANEL.OPANEL	{ EXECUTE CARD
\$IPANEL	IGEOM = 0		
	PRINTS = 1		{ \$IPANEL DATA
	.		
	.		
	.	\$	
\$INLIPS or \$IPATCH or \$TAPEZ or \$ITANK or \$DBBIN			{ input data
@FIN			{ FIN CARD

FIGURE 26 TYPICAL PANEL RUN STREAM.

THETLX(I) array of angles, in degrees, which defines the last points on the ellipses used to describe the body sections.

NN(I) - number of divisions of cross sections desired.

AA(I) - semi-major axes of the defining ellipses.

BB(I) - semi-minor axes of the defining ellipses.

LAST - status flag for merging sections.
 = 1, this section will be merged with the next to form a single section.
 = 3, this section will not be merged with the next.

The cubic patch technique is selected by setting IGEOM = 2 in the \$IPANEL namelist. If the namelist format is selected for cubic patch input, INFORM is set to 2 at the same time. The following is a description of the input required for the namelist input option:

NPTS(I) - an array of four values defining the number of points used for each boundary curve. Maximum of 20 for any curve is allowed.

XW1, YW1, arrays defining the coordinate points of boundary number 1.
 ZW1

XW2, YW2, arrays defining the coordinate points of boundary number 2.
 ZW2

XU3, YU3, arrays defining the coordinate points of boundary number 3.
 ZU3

XU4, YU4, arrays defining the coordinate points of boundary number 4.
 ZU4

NOW - number of divisions in the w direction on the patch into which the patch will be divided for panelling purposes. It should be an odd number. If not, the program will convert it to the next higher odd number.

NOU - number of divisions in the u direction on the patch into which the patch will be divided for panelling purposes. It should be an odd number. If not, the program will convert it to the next higher odd number.

ISOVR - a status flag which controls the joining of more than one patch to form a single section.
 = 0, the current section will not be joined with the last section.

- = 1, the current section will be joined with the last section.
- LAST - a status flag which controls the joining of the current section to the next section.
 - = 0, the current section will be joined with the next section.
 - = 3, the current section will not be joined with the next section.

The TANK option is selected by setting IGEOM = 4 in the \$IPANEL namelist. The following is a description of the input required for \$ITANK namelist:

<u>NAME</u>	<u>DEFAULT</u>	<u>DESCRIPTION</u>
RNOSE	.5	Radius of the spherical nose shape.
NSECN	5.	Number of sections in the spherical nose shape.
RTANK	1.	Radius of the tanks.
LNOSE	3.	Combined length of the nose (includes spherical and transition shapes).
LTANK	10.	Combined length of tank (includes spherical base shape, if any, but not the arbitrary base shape).
NSECB	6.	Number of sections in the base shape.
RAMP	30.	Ramp angle for nose transition shape (used for third order fitted transition shape).
NSECR	3.	Number of sections for nose transition.
ICONE	0.	Cone option flag if ICONE = 1, conical shape will be generated; otherwise a third order transition shape will be generated.
IPRINT	0.	Print flag, if IPRINT = 1, \$OTANK output will be printed.
ISRM	0.	Base section flag. If ISRM = 0, spherical base shape will be generated; otherwise arbitrary base shape will be generated.
NAR	1.	Number of sections in arbitrary base shape.
XA	0.	X-Stations of arbitrary base shape with respect to the base of the tank (excluding spherical base cap).

The TRAPZ option is selected by setting IGEOM = 3 in the \$IPANEL namelist. The following is a description of the input required for \$TRAPZ namelist.

<u>NAME</u>	<u>DEFAULT</u>	<u>DESCRIPTION</u>
NSEC	1.	Number of sections to be generated.
X	0.	X-Stations along the body (usually negative)
YL	0.	Y-Stations for center of lower radius.
ZL	0.	Z-Stations for center of lower radius.
RL	0.	Lower radius.
NRL	7	Number of straight line segments for lower radius.
YU	0.	Y-Station for center of upper radius.
ZU	0.	Z-Station for center of upper radius.
RU	0.	Upper radius.
NRU	7	Number of straight line segments for upper radius.
LINK	0	Flag for linking this component to previous component. LINK=1 Sets status for first point to 1 Otherwise status will be 2.
LAST	3	Flag for linking this component to next component. LAST=0 Sets status for last point to 0. LAST=3 Sets status for last point to 3.
PRINTS	0	Print option flag, if 1, Namelist input and computed values will be printed.

The DBBIN option is selected by setting IGEOM = 5 in the \$IPANEL namelist. The following is a description of the required inputs for \$DBBIN:

DY	Y-Coordinate of the center of the tank pair.
DZ	Z-Coordinate of the center of the tank pair.
SEP	Separation distance between the tank components.
ROT	Angle of rotation measured from the negative Z-axis.
RTANK	Radii of the tank components.

Program Output

The primary output of the PANEL Program is a geometry file in Gentry format, reference 27. Figure 27 illustrates a vehicle configuration which was generated by PANEL. When utilizing the TANK or DBBOUT options, the following data is output to the TZOID, ELLIPS or PATCH generators:

DBBOUT Output Descriptions. -

THO1	Initial angle for generation Body 1.
THL1	Final angle for generation Body 1.
THO2	Initial angle for generation Body 2.
THL2	Final angle for generation Body 2.
DY1	Y-Coordinate of Body 1.
DZ1	Z-Coordinate of Body 1.
DY2	Y-coordinate of Body 2.
DZ2	Z-Coordinate of Body 2.
DDY	Differential Y-Position of bodies 1 and 2.
DDZ	Differential Z-Position of bodies 1 and 2.
TH	Intersection angle of bodies 1 and 2.

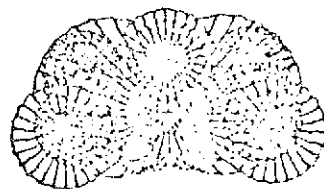
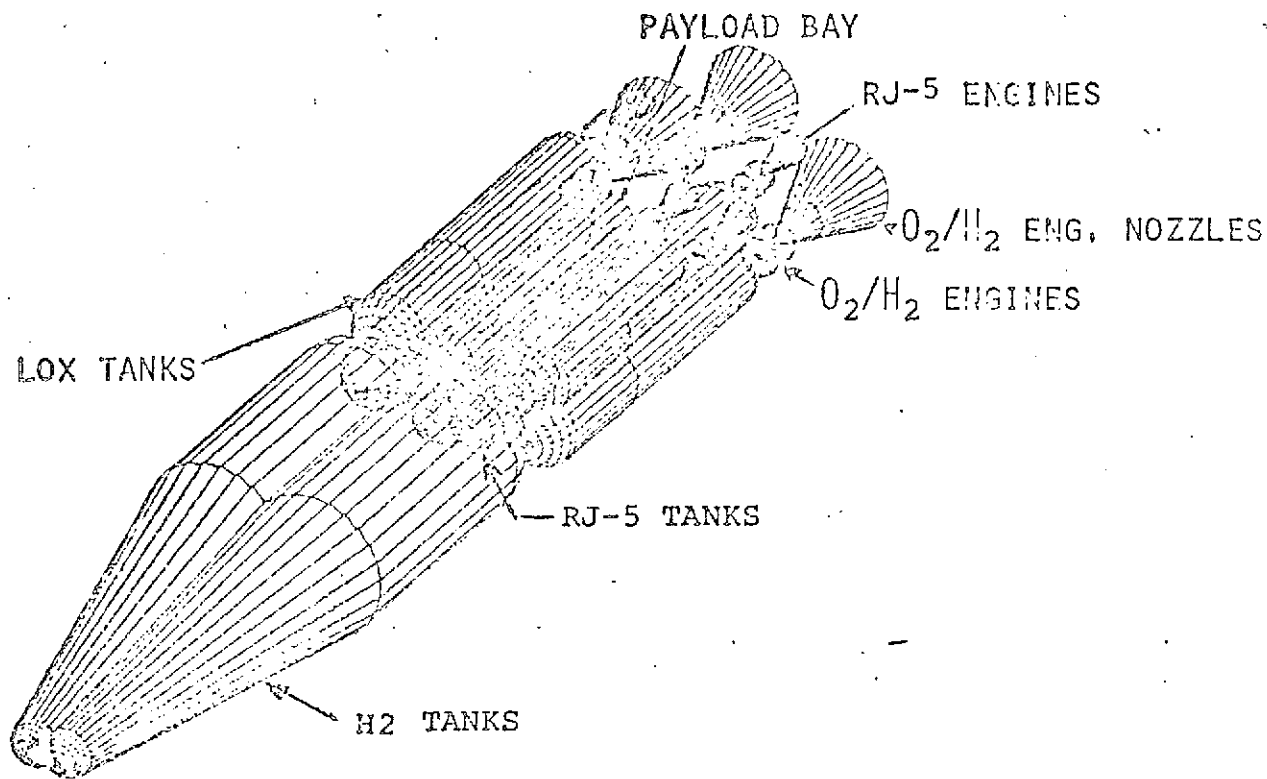


FIGURE 27

INTERNAL ARRANGEMENT - RESEARCH CONFIGURATION

TANK Output Descriptions. -

NSEC	Number of sections generated for the body of revolution.
AX	X-Station generated for the body of revolution.
RAD	Radii generated for the body of revolution.
GAMD	Angle at which transition from a spherical nose occurs - degrees.

PLOTTR: AN INDEPENDENT COMPUTER PROGRAM FOR
THE GENERATION OF GRAPHICAL DISPLAYS.

A computer program is described for generating graphical information from input data or auxiliary analysis programs on a variety of graphical devices. Information can be presented in the usual X-Y plot sense or can be presented as contour plots. The user selects the segments of data to be plotted as well as the scaling, annotation and titling of the resulting plot. Options also exist for tabulating the data in columnar format and for plotting auxiliary text in the vicinity of the plotted information. Display device selection is accomplished by interfacing the basic computer code through routines which convert the internally generated plot vectors to hardware commands for the display device. Separate computer programs are available for the following devices.

Tektronix 4012 Display Terminal

Hazeltine 4000G Terminal (MOPS System)

CALCOMP Drum Plotter

SD4060 Film Plotter

The plotting techniques employed in the computer program are discussed in detail. User's instructions are presented with examples which illustrate the use of the program in generating plotted information from various sources and presenting the information in alternate plot formats.

Program Description

The display of engineering information pertaining to a vehicle design is one of the most important aids in the design analysis process. Historically, design analysis data has been generated on the computer and returned to the engineer in a printed format. The data was normally transformed to a more usable graphical presentation through hand plotting by the engineer or engineering aide. More recently, hardware techniques have been developed which permit automatic plotting of preselected analysis data. The programmer normally provides precoded logic in the technology module to generate plot vector files. The plot vector files are then processed by the hardware plotting device and the graphical display generated is returned to the engineer. The classical approach described above requires that the plot vector file generation software be imbedded in every technology program.

The development of the multiple program technological analysis techniques of the EDIN system has significantly increased the requirements for the presentation of analysis data of an interdisciplinary nature. The use of plot vector generation code within the technology modules does not meet the interdisciplinary or the comparison requirements. The objective in the development of a plot program, which is independent of the technology programs, is to provide a means of plotting any data which was previously generated by a technology module and stored in a data base.

The plot program specifications require the generation of X-Y plots, cross plots and contour plots which compare not only data from a specific analysis or any individual programs but also to provide comparisons with previously generated data from any source. The independent plot program developed meets the desired specifications or objectives. It assumes no prior knowledge of the information that is to be plotted at the time of execution. Plot scales, annotations, titles, etc. are described by the user through input. Plot data requests are specified by the user and the data can come from many sources.

The independent plot program was an evolutionary development based on requirements which were developed through use of the multiple technology analysis technique. It has served well during the period of development and seems to be adequate for most technological data sources.

The plot instructions are read from the input data file and the data to be plotted can be read from either the input file or from auxiliary data files. The input instructions are read using a single NAMELIST input list.

\$PLOTIN.....\$

Default values are preset for all input variables. The default values tend to minimize the amount of user input required to generate a plot. Generally the input specifications are patterned after the procedure one might follow in preparing plot by hand. The user can select such options as grid, axis generation, annotation, titles, auxiliary text, line type, symbol specifications, etc. Those options not specifically selected by the user are generally bypassed in the program.

A simple self tutoring procedure for a new user can be described as simply to scan the input specifications in a later section and include values for the input variables which require changing. A discussion of the plotting options which are available is presented in the following paragraphs.

Plot Data Input Option

The data to be plotted is read into the program in either of two formats, array format or observation format. The data may come from the normal input or from a binary file in accordance with the following specifications:

INPUT = 0 Data will be loaded directly into the OBSTH array from the normal input unit.

INPUT = n n Specifies the logical unit number from which the OBSTH array will be loaded.

Array Format. - Array format specifies the numerical values to be plotted are arranged in groups of similar observations such as time, attitude or velocity. Mathematically, array format is the sequence of numerical values:

$$\text{OBSTH}_{ij}; i = 1, \text{NUMP}; j = 1, \text{NOBSER}$$

where NUMP is the number of observations and NOBSER is the number of observation functions. The plot data is actually a single dimensional array OBSTH_k where k is the array element defined as:

$$k = (j - 1) \text{NUMP} + 1$$

The independent plot program normally reads data from the input file in array format but other options are also available.

Observation Format. - Observation format specifies the numerical values are arranged in groups called observations. Each observation represents a sequence of values defining one element in each plot array. Observation format may be expressed mathematically as:

$$\text{OBSTH}_{k(i,j)}; i = 1, \text{NOBSER}; j = 1, \text{NUMP}$$

where NOBSER is the number of observation functions and NUMP is the number of observations. k is the array element defined as:

$$k(i,j) = (j - 1) \text{NOBSER} + 1$$

The independent plot program has the capability of reading data as observation format when the alternate value variable:

$$\text{ALTVAL} = \text{.TRUE.}$$

option is specified. If specified, a transposition of the observation format to array format is performed and the array format ultimately overwrites the input values of OBSTH.

Alternate Data File. - In addition to the two formats which can be selected for reading from input, an alternate data file can be specified:

INPUT = n

When this option is specified, the plot data is read from the logical unit n in observation format, transposed and placed into the OBSTH array in accordance with the specified values of NUMP on NOBSER. No file positioning is done by the program but the user can manipulate the file with the following input variables:

REWIND = .TRUE. Rewinds n.

NSKIPF = m Skips forward m Fortran files.

NSKIPR = m Skips forward m Fortran records.

The alternate file format may include as the first record one word specifying the number of records on the file. If the one word record exists, the plot program can read it and store the value as NUMP. This option is activated by the specification:

NUMP15 = .TRUE.

Multiple cases may be executed for as many plot cases as desired. Therefore, different options may be specified on successive cases. On the last case, the parameter:

STOP = .TRUE.

is set which causes program termination. No other plot instructions are executed in the last case.

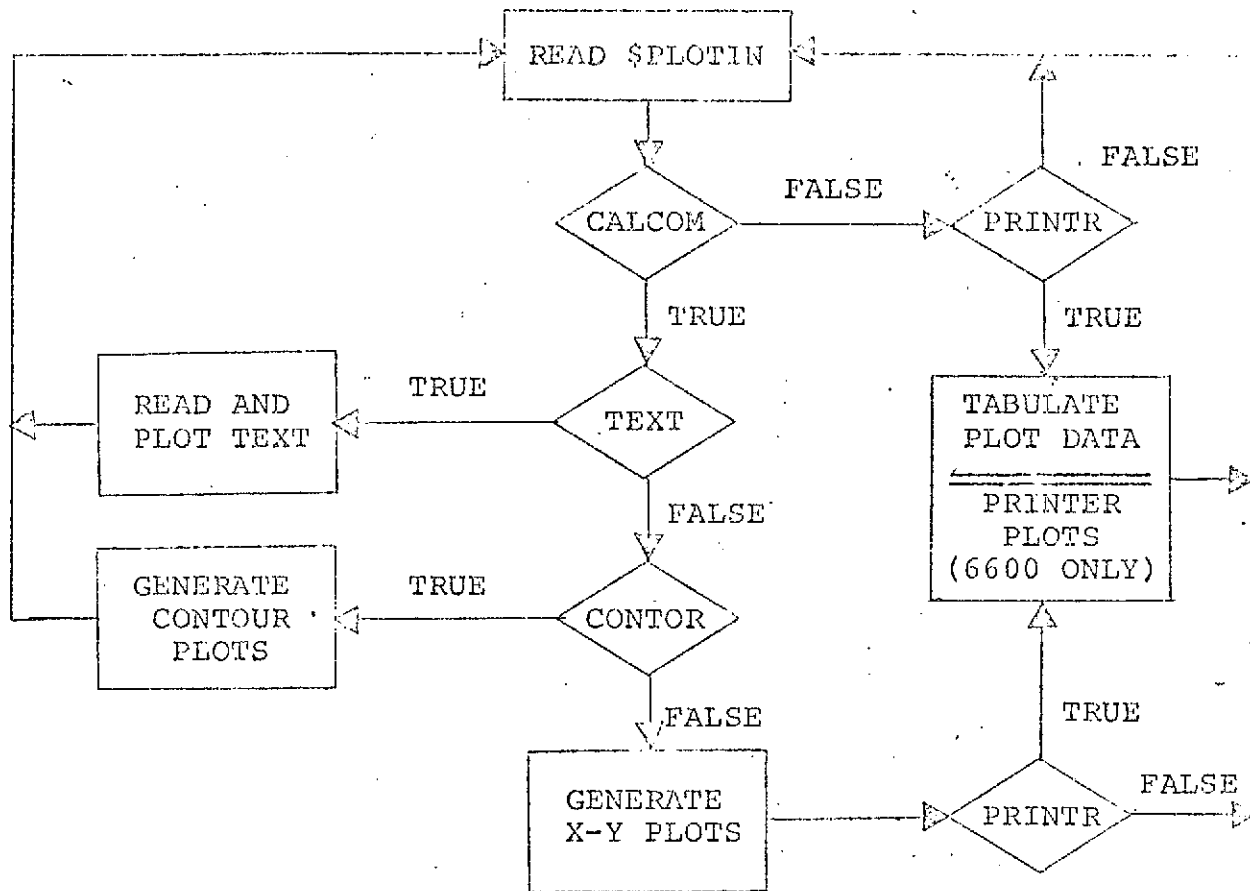
X-Y Plots

The PLOTTR program provides for the display of numerical information of the form:

$$Y_i = f_j(X_i); i = 1, NUMP; j = 1, NPLOT$$

where X and Y are arrays of observations of known length, NUMP. Any array may be plotted with respect to any other array and any number of pairs may be presented on a single plot. NPLOT is the number of PLOTS desired. Figure 28 illustrates the display of two multiple curve plots. The input data for this plot is discussed in a later section.

Plotting is specified in accordance with the input variable array:



(A) PROGRAM SCHEMATIC.

DESIRED OUTPUT	INPUT VARIABLE OPTION			
	CALCOM	CONTOR	TEXT	PRINTR
X-Y PLOTS	TRUE	FALSE	FALSE	T or F
CONTOUR PLOTS	T or F	TRUE	FALSE	T or F
PLOT TEXT	TRUE	FALSE	TRUE	NA
TABULATION	T or F	T or F	FALSE	TRUE

(B) OPTION SPECIFICATION.

FIGURE 28 PLOT OPTIONS.

OBSPLT = $K_1, K_2, K_3, \dots, K_n$

Each K is a packed integer which defines the array pair in the OBSTH array to be plotted:

$K_i = \underline{XXYY}$

where XX represents a two-digit sequence defining the array to be plotted as the x-coordinate. YY represents a two-digit integer sequence defining the y-coordinate of the plotted line. Any number of K values may be specified, each representing a line on the plot. A plot sequence is terminated by a value:

$K_i = 0$

A second plot sequence may be specified by simply adding values of K:

$K_{i+1} = \underline{XXYY}$

The last value of K should be:

$K_n = 7777$

which terminates the plotting and returns program control for new input data.

Plot Positioning. - The PLOTTR program has input parameters which control the defining of a new frame and the position of the plot within the frame. The parameters:

DXG and DYG

define the X- and Y- coordinates of the lower left corner of the plot in inches with respect to the lower left corner of the current frame. The parameters:

XMOVE and YMOVE

define the coordinates of the lower left corner of the next frame with respect to the lower left corner of the current frame. The frame includes all physical plotting which takes place as a result of a single set of plot instructions (defined by \$PLOTIN). Frames of data may be superimposed to accomplish certain analysis objectives.

Line Type and Symbols. - The parameters, LINTYP control the type of lines that are to be drawn between the data points. The magnitude determines the frequency of symbols and the sign determines the combination of lines and symbols.

LINTYP = n Means a symbol is to be drawn every nth point.

If n > 0, Lines and symbols are drawn.

If n < 0, Only symbols are drawn.

The parameter INTEQ determines which symbol is to be used by the specification of an integer from 1 to 22. A value of 0 specifies no symbols.

Data Scaling. - Data arrays can be scaled absolutely in terms of the axis length on which they are plotted or the arrays can be scaled relative to one another. The parameters:

XSIZE and YSIZE

specify the x- and y- axis lengths in inches. The first curve specified (see OBSPLT) determines the scale factor and the relative starting position for all curves on the plot. The array:

SCALEF_i

specifies individual scale factors for each plot array. This allows meaningful comparisons of multiple curve plots where the range of the data array differs significantly.

Elimination of automatic scaling may be specified in either or both directions by the specification:

MYX = .TRUE.

and/or

MYX = .TRUE.

If the MYX option is specified, the user must specify the scale factor and starting value the X-axis:

SCALEX = units/inch

STARTX = inches

If the MYX option is specified, the user must specify the scale factor and starting value of the Y-axis.

SCALEY = units/inch

STARTY = inches

The scale factors and start values are set by the program each time automatic scaling (MYX and MYX = .FALSE.) is specified so the scaling from previous cases may be used by properly setting the option flags MYX and MYX.

Scale Annotation. - Under the automatic scaling option the program generates axes of the specified length with tick marks at one inch intervals. The tick marks are identified by numerical values below or to the left of the axis centered on the tick. The axes may be notated additionally by a 6-character input name centered on the axis. One name may be input for each observation function as follows:

OBSERV_i = N_i, i = 1, NOBSER

where N_i is a hollerith name of the form, nH name, and NOBSER is the number of observation functions.

Grid Generation. - A grid may be generated which represents vertical and horizontal lines at the tick marks by the input parameter:

GRID = .TRUE.

Title Generation. - A title from the plot may be generated by setting the parameter:

NREM = n

where n is the number of words of the title. The program will use the first n words of the REM array.

REM = N_i; i = 1, NREM

and place this title at the top of the plot. The height of the characters is in the title as specified by the parameter REMSIZ in inches.

Auxiliary Plot Text

The plotter program has the capability of generating auxiliary plot text by a separate case setup as follows:

\$PLOTIN TEXT = .TRUE.,\$
(text cards)

The text cards immediately follow the case data. The character strings given on the text cards are reproduced exactly starting the first card at a location specified by:

DXG and DYG

in inches. The character height specification is given by HTEXT. Cards will be read and the characters plotted with line spacing of:

$1.5 * HTEXT$

until a card is encountered with a numeric 2 in column 1. Control is then returned for a new case.

Virtual and Display Window

Virtual and display graphics deals with the translation of the user's data to a physical location on the display device. The virtual space may be of any dimension while the display space is limited by the physical size of the display device. The function of the virtual graphics package is to map the virtual space into the specified display area. With an understanding of the relationship between the virtual space and the display area, the user can freely manipulate the display to reflect his need. For example, he can plot three different sets of data in the same display area or he can display the same data to different scales to meet special needs.

The Virtual Window. - The user defines the scale (SCAL) of the virtual window in inches and the position (DXG and DYG) in inches with respect to the plot device origin. Graphic lines (vectors) and portions of lines which lie outside the virtual window are automatically eliminated or clipped by the windowing routines, while those which lie within or pass through the window are scaled and fitted into the appropriate portion of the display.

The user's data area may be conceived of as existing virtually within the computer, and is analogous to a coordinate system with infinite dimension. The unit of measurement in virtual space is arbitrary and representative of any unit the user wishes, from milligrams to light years. In the case of the PLOTTR program, the virtual window is established by XSIZE and YSIZE and includes the plot area itself plus some margin on all sides for plot annotation. In actuality, the virtual window is a square area whose sides are related to the larger of XSIZE or YSIZE. In effect, the user constructs the desired plot in virtual space and the virtual graphics package scales the data to fit into the

desired display area. Figure 29 illustrates the use of virtual graphics. The heavy outside border is the window which is scaled to the specified display size SCAL.

ORIGINAL PAGE IS
OF POOR QUALITY

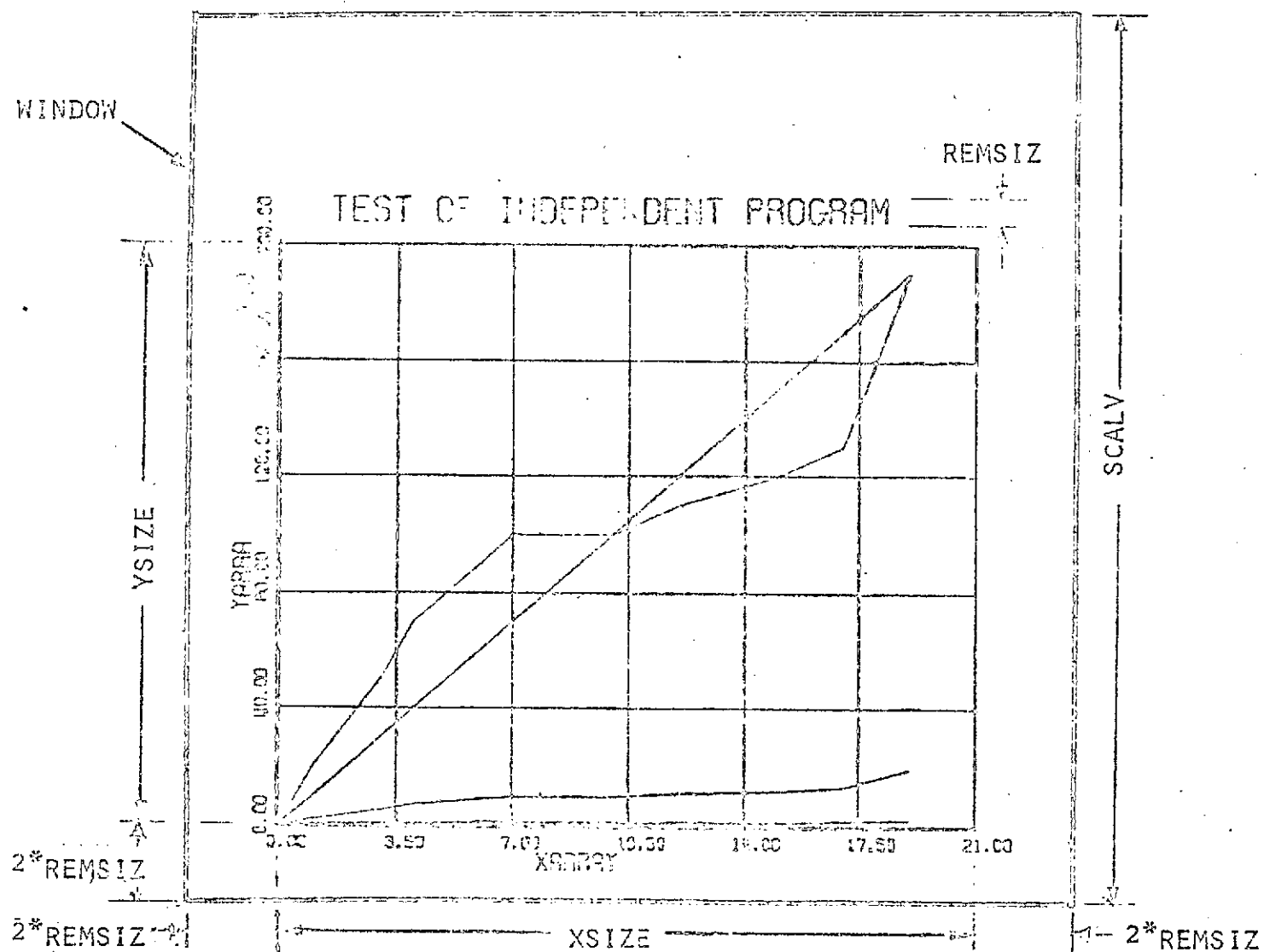


FIGURE 29 ILLUSTRATION OF VIRTUAL GRAPHICS.

Contour Plots

PLOTTR contains a simple and rapid code for producing contour plots of three dimensional functions. The function must be specified in the form:

$$Z_{ij} = F(X_i, Y_j); i = 1, 2, \dots, M, j = 1, 2, \dots, N \quad (1)$$

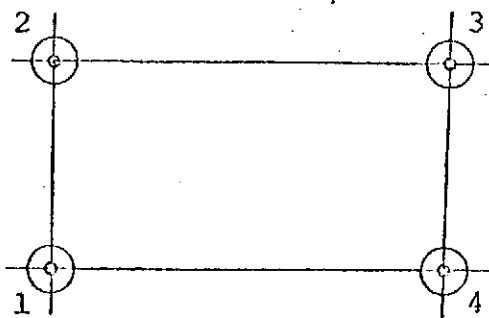
That is, the function must be defined over a regular grid in the X-Y plane. Contour plots are produced on the target display device but cannot be generated as printer plots. The resolution of the contours is dependent upon the mesh increments in X and Y.

Figure 30 illustrates the type of results which can be expected from the CONTOR option. The illustrated contour data has a "resolution" of 0.1 in X and Y.

Generation of Contour Vectors. - Suppose a function of two independent variables is defined over the regular mesh X_i, Y_j . Then $(M-1)*(N-1)$ rectangular boxes can be selected from the adjacent points:

$$P = (X_i, Y_j), (X_i, Y_{j+1}), (X_{i+1}, Y_{j+1}), (X_{i+1}, Y_j) \quad (2)$$

Let the corners of any such box be identified in a clockwise manner as illustrated below:



RECTANGULAR ELEMENT CORNER IDENTIFICATION.

Given such a rectangle, the corner points and the function values at these corner points may also be uniquely defined by the rotation shown below:

SONIC BOOM OVERPRESSURE CONTOURS

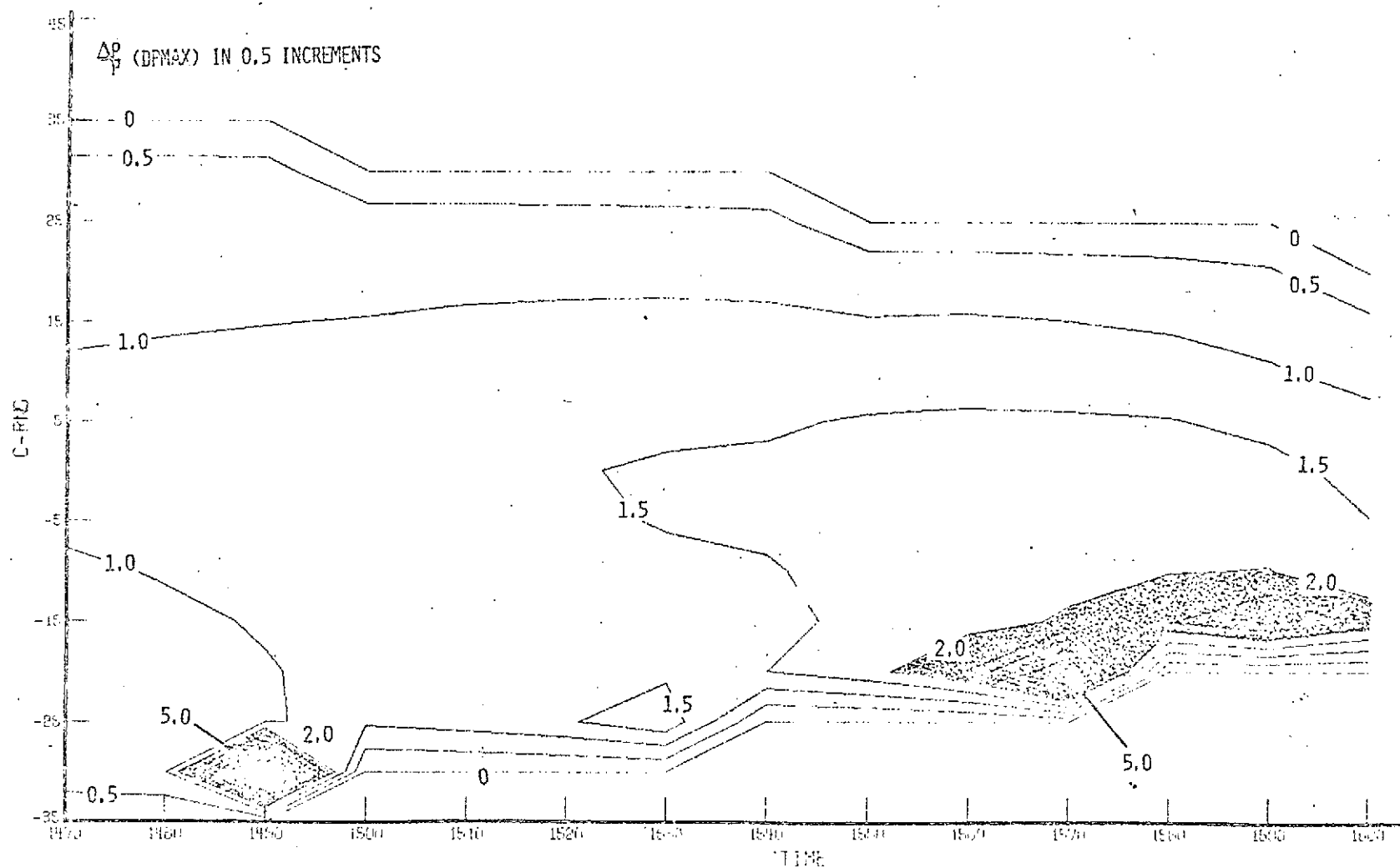
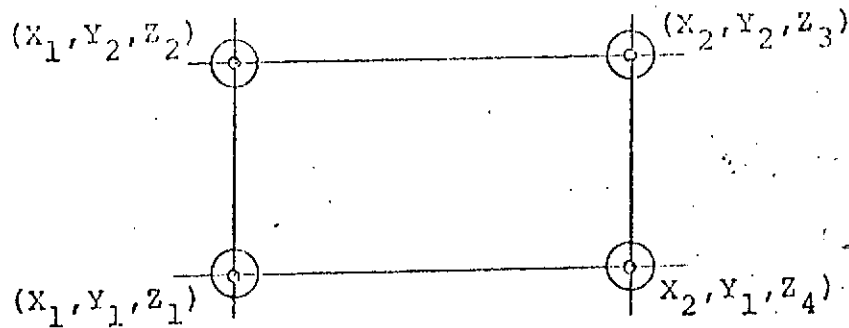


FIGURE 30

ILLUSTRATION OF CONTOUR PLOT DATA OUTPUT.

ORIGINAL PAGE IS
OF POOR QUALITY



LOCAL COORDINATE AND FUNCTION VALUES

Now consider the possibility that a contour of value $Z = Z'$ passes through a given elemental rectangle. First, transform the corner values of Z by subtracting Z' to give the modified corner values:

$$\bar{Z}_1 = Z_1 - Z'$$

$$\bar{Z}_2 = Z_2 - Z'$$

$$\bar{Z}_3 = Z_3 - Z'$$

$$\bar{Z}_4 = Z_4 - Z'$$

(3)

Examining the topology of the contour line trace across the elemental rectangle, it follows that sixteen (16) possible types of contour trace exist. For each modified corner value given by three (3) is either greater than or equal to 0, or less than zero, giving 2^4 topological types of trace. These trace types may be uniquely identified by a four digit binary number whose elements sequentially correspond to the corner points in the clockwise sequence of figure 31, where 1 signifies that $\bar{Z}_i > 0$. These sixteen types of trace are illustrated in figure 31 with their corresponding four (4) digit binary number and contour trace type.

It can be seen that only two (2) of the element topologies result in no contour trace. Two element topologies result in two contour traces across the element. (These contour traces may be of either form displayed in figure 31 for $I = 1010$ or $I = 101$.) Assuming linear interpolation for Z along the elemental rectangle sides and local straight line contour traces within the element, the end points of the contour trace are readily found to be given by the expressions such as:

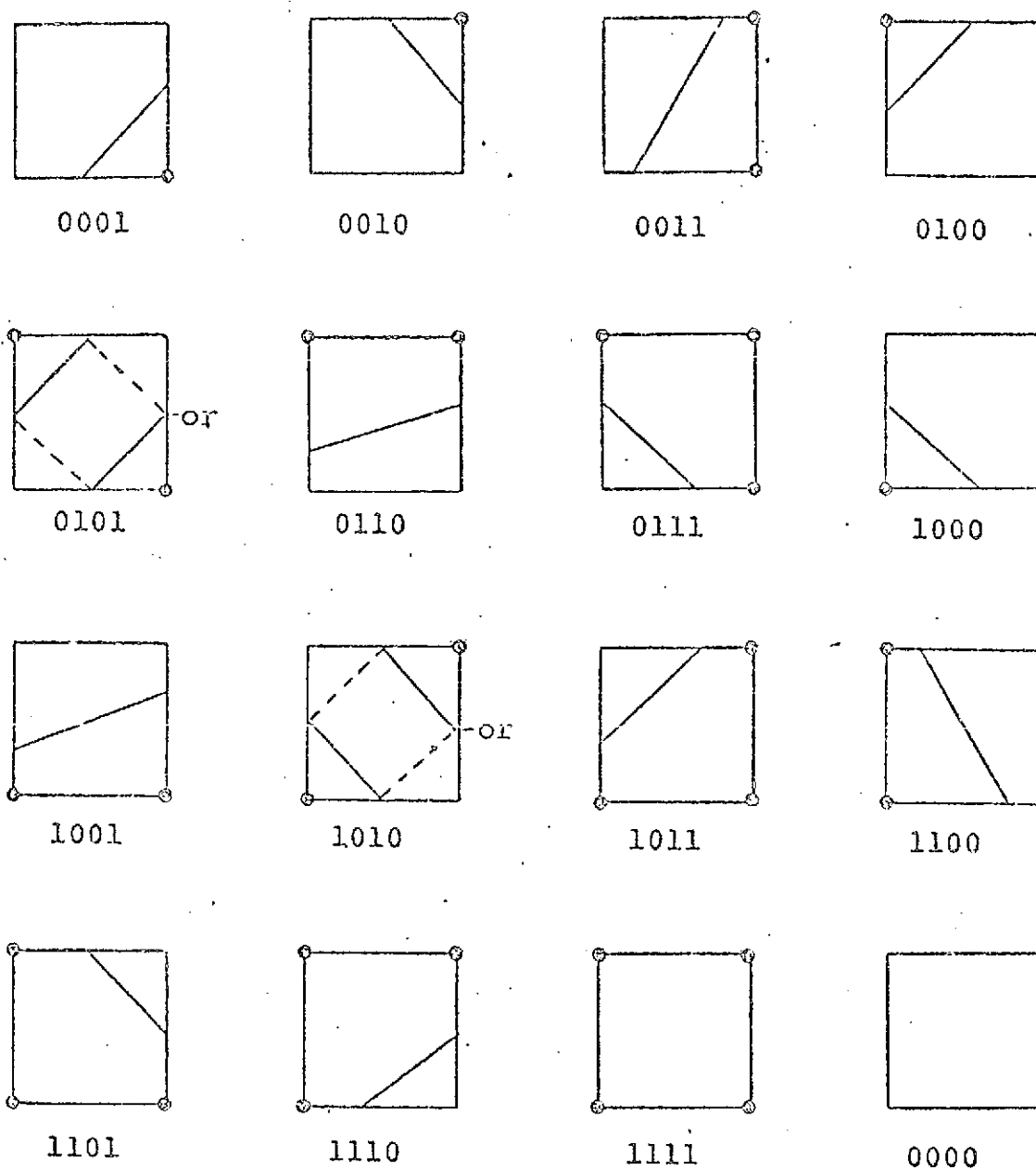


FIGURE 31 RECTANGULAR ELEMENT BINARY CODE
AND CONTOUR TRACE TOPOLOGIES.

$$\underline{I = 1010}$$

$$X_{e_1} = X_1$$

$$Y_{e_1} = Y_1 + F(Y_1, Y_2, Z_1, Z_2) \quad (4)$$

$$X_{e_2} = X_1 + F(X_1, X_2, Z_3, Z_4)$$

$$Y_{e_2} = Y_2$$

and

$$\underline{I = 1110}$$

$$X_{e_1} = X_1 + F(X_1, X_2, Z_1, Z_4)$$

$$Y_{e_1} = Y_1 \quad (5)$$

$$X_{e_2} = X_2$$

$$Y_{e_2} = Y_1 + F(Y_1, Y_2, Z_4, Z_3)$$

Where (X_{e_1}, Y_{e_1}) and (X_{e_2}, Y_{e_2}) are the contour trace end points and the function F is defined by:

$$F(A, B, C, D) = (B - A) \cdot \left| \frac{C}{D - C} \right|$$

Two Ambiguous Cases. - Two ambiguous cases arise for $I = 1010$ and $I = 101$. The contours may then be of either the solid or dotted line type in figure 32. The ambiguity is resolved when the "fold diagonal" is defined. Thus when $I = 0101$, if the principal diagonal is the fold diagonal, the dotted lines supply the correct contour types, if the other diagonal is the fold line, then the solid lines are the correct contour types. The converse is true when $I = 1010$.

Definition of the fold line requires more information than is available within a single elemental rectangle. Consider the case $I = 1010$ in the illustration below. In (a) the upper right and

lower left high elemental rectangle regions indicate a principal diagonal fold. In (b) the lower upper left and lower right elemental rectangle regions illustrate a fold about the other diagonal.

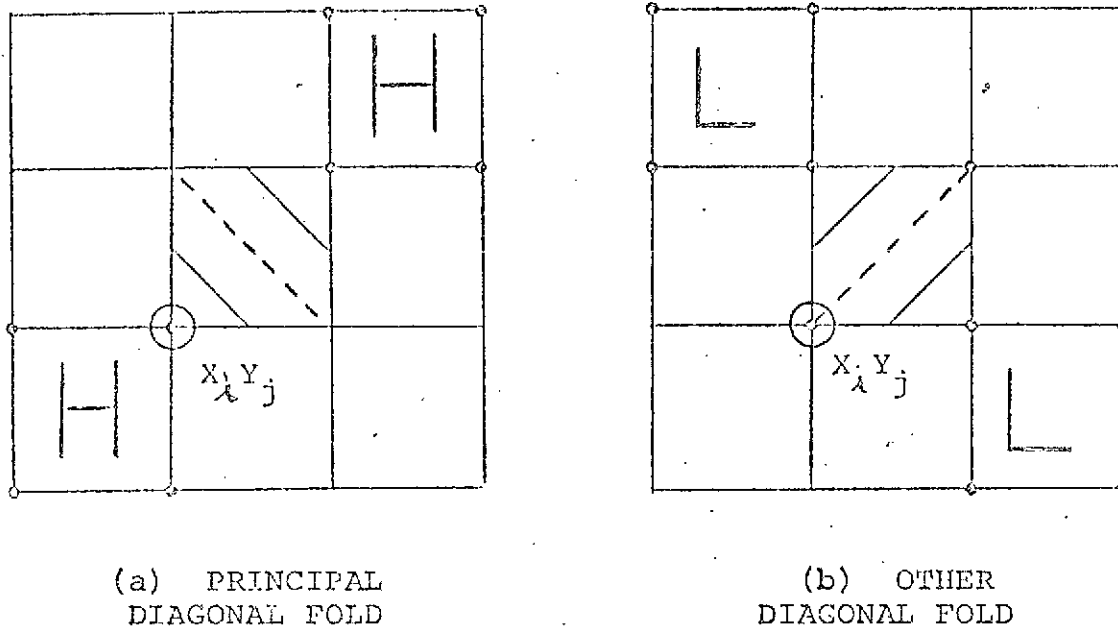


FIGURE 32 TWO ANALOGOUS CASES.

The program contains logic for resolving the ambiguous cases using the above technique. A weighted assessment of the probability of each diagonal by the fold line is incorporated within the code. This logic covers the possibility of any adjacent elemental rectangle being absent due to edge or corner conditions in the Z_{ij} array.

Loading the Data. - The contour data is loaded into the OBSTH array as follows:

$OBSTH_k; k = 1, NUMP * NOBSER$

where k is defined as:

$k = (j - 1) NUMP + i$

$i = 1, NUMP$

$j = 1, NOBSER$

Data may also be loaded from an alternate input file using the INPUT parameter as described earlier. Two additional arrays defining the x and y mesh parts must be loaded as:

$\text{XMESH}_i; i = 1, \text{NUMP}$

$\text{YMESH}_j; j = 1, \text{NOBSER}$

The values must correspond to the function values loaded in the OBSTH array.

Contour Definition. - The contours desired are specified by the user using the following input:

$\text{NZCUTS} = n$

where n is the desired number of contours. The array:

$\text{ZCUTS}_i; i = 1, \text{NZCUTS}$

defines the individual contour values.

Title and Axis. - Title and axis options are the same as described earlier for x-y plots.

Program Usage

The program can be used with two online and two offline display devices as follows:

<u>DISPLAY DEVICE</u>	<u>TYPE</u>	<u>PROGRAM NAME</u>
TEKTRONIXS	Online	*TEKTRON.OPLOT
MOPS	Online	*HAZEL.OPLOT
CALCOMP	Offline	*CALCOMP.OPLOT
SD4060	Offline	*SD4060.OPLOT

*File qualifier is EX42-00002.

Each device uses a different program. The data setup is identical so far as parameter names are concerned. However, the desired scaling value may be different because of the size of the available plotting area for the different devices.

Each plot interface has special instructions that are particular to the plot device. For example, the CALCOMP program requires the mounting of a physical tape which must be named 19:

```
@ASG,T 19,8C, CAL
@XQT EX42-00002*CALCOMP.OPLOT
(Data)
```

The TEKTRONIX program requires the user to input the terminal line speed and the maximum number of vectors per buffer load:

```
@XQT EX42-00002*TEKTRON.OPLOT
30 18
(Data)
```

The above example is for 30 characters per second line speed and 18 vectors per buffer load. No more than 18 vectors should be specified.

The MOPS program requires the use of the function code 0 above the keyboard before data is entered:

```
@XQT EX42-00002*HAZEL.OPLOT
Depress FC-0
(Data)
```

No special input is required to use the SD4060 program:

Program Input

The basic input to PLOTTR is the \$PLOTIN namelist specification. All data except auxiliary text may be read in this form if desired; but other forms of input may be activated in accordance with the options specified in \$PLOTIN.

The use of NAMELIST input was chosen for the following reasons:

1. It is a simple name oriented input easily understood by most computer users.
2. The format is standard and does not require relearning from program to program.
3. It is easily modified by the engineer or programmer when adding input variables to the program.

When a NAMELIST read is encountered in the program, the entire input file is scanned up to an end-of-file or a record with a dollar (\$) in column 2 followed by the NAMELIST name requested by the program. No imbedded blanks may be used. Succeeding data items are read until a second dollar is encountered signifying the end of the NAMELIST input. Any data on the input file before the requested NAMELIST is found will be ignored. All data between the opening and closing dollar is interpreted by the NAMELIST input routine. The data item within the NAMELIST statement may be in either of two forms:

$$V = C,$$

$$A(n) = D_1, \dots, D_m,$$

V is a non-subscripted variable name. C is a constant which may be real, integer, logical (T or F) or hollerith (lHname). A is an array name and n is an integer constant subscript, D_1, \dots, D_m are simple constants or repeated constants of the form $k \cdot C$, where k is the repetition factor. Data items and constants must be separated by commas. For a subscripted array name, the number of constants need not be equal but may not exceed the number of array elements needed to fill the array. More than one card may be used for input data and arrays may be split between cards. All except the last record must end with a constant followed by a comma and no sequence numbers may appear. The first column of each record is ignored. The

set of data items may consist of any subset of the variable names associated with the NAMELIST name and the name need not be any particular order. More details on the use of NAMELIST are available in any FORTRAN users guide, but the above description should be sufficient for the operation of the PLOTTR program.

\$PLOTIN Namelist Data. - This section contains an alphabetical list of all input variables in the \$PLOTIN namelist set. The default value and a brief description of the usage of each variable is given.

<u>Name</u>	<u>Default(s)</u>	<u>Description of Input</u>
ALTVAL	.FALSE.	Logical variable. If .True. data will be read as observation functions. Otherwise data will be read as plot arrays.
BLOCK	.FALSE.	Logical variable. If .True., binary blocking of the observation unit will be expected. (CDC 6600 only)
CALCOM	.TRUE.	Logical variable. If .True., x-y plots will be generated. This variable is used to activate any device to which the program is linked such as Tektronics, SD4060 or Varian.
CONTOR	.FALSE.	Logical variable. If .True., a contour plot will be generated from OBSTH data CALCOM must be set .True.
COPY	.FALSE.	Generate hard copy (online devices).
DIALOG	.FALSE.	Logical variable. If .True., data base output routine will be called.
DXG	1.0	X-axis origin for the current chart relative to the current device origin specified by XMOVE.
DYG	1.0	Y-axis origin for the current chart relative to the current device origin specified by YMOVE.

<u>Name</u>	<u>Default(s)</u>	<u>Description of Input</u>
GRID	.FALSE	Logical variable. If .TRUE., one inch grid will be generated on x-y plots.
HTEXT	.21	Height of title in inches.
INPUT	0	Zero for reading plot data from cards .GT.0 for reading plot data from another unit.
INTEQ	1	Integer from 0 to 22 indicating the plot symbol.
LINTYP	0	Control parameter which describes the type of line to be drawn through the data points. The magnitude determines the frequency of plotted symbols. I.E. LINTYP=4 Means every fourth point. LINTYP=0 Straight lines with specified symbol at the end of the line (see INTEQ). LINTYP=+ Lines and symbols. LINTYP=- Symbols only.
MYX	.FALSE.	Logical variable. If .TRUE., user may input STARTX and SCALEX.
MYY	.FALSE.	Logical variable. If .TRUE., user may input STARTY and SCALEY.
NAMES (100)	.FALSE.	Logical variable. If .TRUE., NOBSER six-character names will be read. These are the plot array titles.
NAXIS	1	Number of times the beam or pen will trace the x- and y- axes.
NDECP	2	Number of decimal places used in scale annotation.
NOBSER	NONE	Number of observation functions (or plot arrays).
NPAGE	0	Page number of the plot (printer only).
NREM	0	Number of words of title to be read. If not zero, NREM 10-character words will be read immediately after the \$PLOTIN namelist.

<u>Name</u>	<u>Default(s)</u>	<u>Description of Input</u>
NSKIPF	0	Number of observation function files that will be skipped on observation unit before starting to read data from it.
NSKIPR	0	Number of observation functions to skip on observation unit before starting to read data.
NUMP	NONE	Number of plot points per plot array or the number of observations. There is an internal limit of 213 points.
NUMP15	.FALSE.	Logical variable. If .TRUE., NUMP will be read from observation unit as the first record.
NZCUTS	NONE	Number of contours requested.
OBSERV (100)	BLANK	Observation function names to be used for scale annotation. They are read in namelist format as: OBSERV=nHname1,nHname2,----.
OBSPLT (120)	7777	Integer definition of the plot functions. Each pair of plot arrays or observation functions is defined by a 4-digit number, the first two represent the independent variable, the second two represent the dependent variable according to the input order of the plot arrays or OBSTH functions. A zero indicates the end of one chart. More charts can be generated up to a limit of 120 entries in the OBSPLT array. User should enter a value of 7777 after the last chart.
OBSTH (2000)	NONE	Plot data in one of the following formats. For ALTVAL=.FALSE., plot arrays are loaded, A(1),--,A(N),B(1),--B(N),C(1)--C(N), For ALTVAL=.TRUE., plot arrays are loaded, A(1),B(1),C(1),-----,A(N),B(N),C(N), NOTE: The maximum number of plot points is 2000 but can be changed to the alteration of 3 Fortran statements in main program as follows:

<u>Name</u>	<u>Default(s)</u>	<u>Description of Input</u>
		COMMON/AESOPD/ADATA(n+1)
		REALOBSTH(n)
		DATA NADATA/n/
		The value of n may be set to any desired value. Program load size is altered in direct relation to the number n.
PAGE	.FALSE.	Start a new frame at XMOVE, YMOVE from previous origin.
PRINTR	.FALSE.	Logical variable. If .TRUE., tabulation will be generated.
PRINTO	.FALSE.	Logical variable. If .TRUE., the namelist input data will be printed.
REM (30)	BLANK	Title to be placed at the top of the chart. It is read in namelist format as: REM=nH title of plot,
REMSIZ	0.21	Height of title in inches.
REWIND	.FALSE.	Logical variable. If .TRUE., observation unit will be rewound before reading it.
SCAL	7.5	Size of the plot device window in inches. Virtual plot specified by the maximum of XSIZE and YSIZE will be scaled to this dimension.
SCALEF (100)	1.0	Scale factor array. One for each plot array or observation function. It is used to scale one plot array relative to the others for plot purposes but does not affect the original data in OBSTH.
SCALEX	1.0	Units per inch for X.

<u>Name</u>	<u>Default(s)</u>	<u>Description of Input</u>
SCALEY	1.0	Units per inch for Y.
STARTX	0.0	Starting value for X-axis.
STARTY	0.0	Starting value for Y-axis.
STOP	.FALSE.	Logical variable. If .True., program will stop without generating additional plot information.
TEXT	.FALSE.	Logical variable. If .True., card images will read and plotted scaling will be in accordance with the following formula: $6*NREM*HTEXT/SCAL$
XMESH (100)	None	Array of points defining the X-axis of a contour plot.
XMOVE	8.5	X-distance between plot origins. For on-line devices, a screen erasure is affected.
XORIGN	1.0	X-axis origin for the current chart relative to the current device origin specified by XMOVE.
XSIZE	6.0	X-size length in inches for virtual plot. It also specifies number of scale (and grid) divisions which will be employed. Origin is moved before plotting if the PAGE option is invoked.
YMESH (100)	None	Array of points defining the Y-axis of a contour plot.
YMOVE	0.0	Y-distance between plot origins. For on-line devices, a screen erasure is affected.
YORIGN	1.0	Y-axis origin for the current chart relative to the current device origin specified by YMOVE.

<u>Name</u>	<u>Default(s)</u>	<u>Description of Input</u>
YSIZE	8.0	Y-axis length in inches for virtual plot. It also specifies number of scale (and grid) divisions which will be employed. -Origin is moved before plotting if the PAGE option is invokled.
ZCUTS (25)	None	Values of the contours.

Physical Characteristics

The basic PLOTTR program is written in Fortran language and interfaced to the four plot devices as illustrated in figure 33. The control program subroutines reference the CALCOMP subroutines AXIS, SCALE, NUMBER, SYMBOL and LINE. These routines have been modified slightly to call PLTT (rather than PLOT). All display device interface is accomplished through the single driver PLTT. PLTT has three basic entries:

Initialization

Plotting

Termination

which perform special functions depending upon the device interfaced. The windowing and clipping package is called by PLTT at the "plotting" entry. The data is scaled to the specified window size and vectors outside the window are deleted or modified.

Program Loading

The program is overlayed to execute in approximately 20,000 decimal but the size varies with the interface software requirements. The four program sizes are shown below:

TEKTRONIX	19400
CALCOMP	20100
SD4060	21400
MOPS	18700

The construction of the program requires the following control cards:

```
@QUAL EX42-00002
@COPY,R * PLOTTR
@MAP file .OPLOT,OPLOT
```

The basic plot software resides on the file EX42-00002*PLOTTR. There is a file containing interface software for each hardware device. The file used depends upon the interface. Map directive elements are also stored on the interface files:

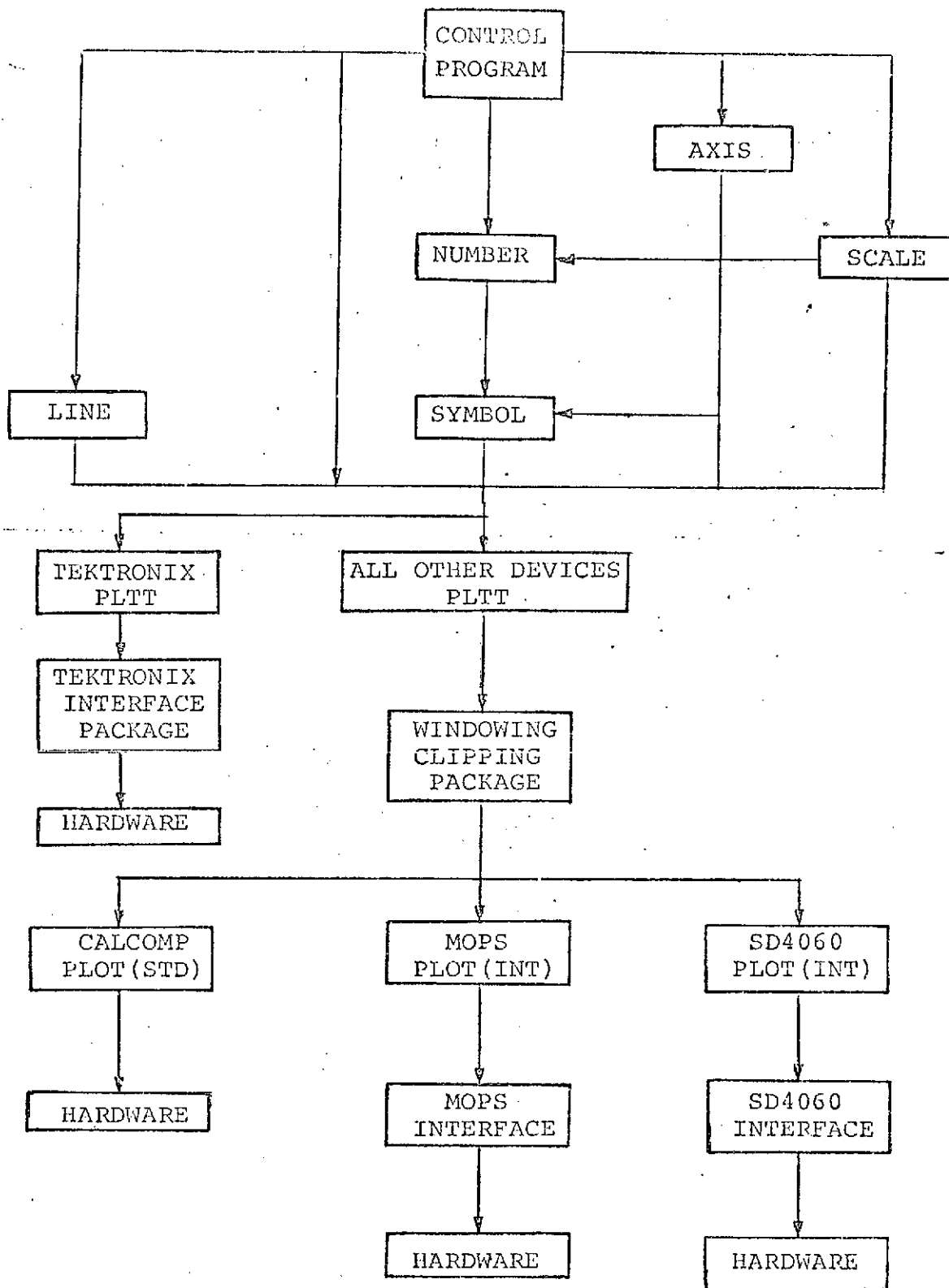


FIGURE 33 PLOTTR INTERFACE.

TEKTRONIX	EX42-00002*TEKTRON.OPLOT
CALCOMP	EX42-00002*CALCOMP.OPLOT
SD4060	EX42-00002*SD4060.OPLOT
MOPS	EX42-00002*HAZEL.OPLOT

The TEKTRONIX map directives are illustrated below:

```

LIB *TEKTRON.,*TEKLIB.
LIB *TEKTRON.,*TEKLIB.
NOT TPF$.PPLNLN
NOT TPF$.NOGRAF
NOT TPF$.DMPBUF
NOT TPF$.XYCNUT
NOT TPF$.MOVEA
NOT TPF$.DRAWA
NOT TPF$.VECMOD
NOT TPF$.SCRENE
NOT TPF$.HDCOPY
NOT TPF$.PLTW
NOT *TEKLIB.CHARS
NOT *TEKLIB.CHARS/TEKPLOT
NOT *TEKLIB.TKCH
NOT IMAGE
SEG MAIN
IN PLOTTR
SEG AA*
IN CONFLT
SEG BB*.(MAIN)
IN THROBS
SEG BB1*. (BB)
IN HPLNLN
SEG BB2*. (BB)
IN PPLNLN
END

```

Map directive elements for the other three device interfaces are shown in figure 34

```

1:LIB *CALCOMP, MSC*LOCALIB, MSC*PLTLIB
2:LIB MSC*LOCALIB, MSC*PLTLIB, *CALCOMP
3:NOT IMAGE
4:SEG MAIN
5:IN PLOTTR
6:SEG AA*
7:IN COMPLT
8:SEG BB*, (MAIN)
9:IN THRODS
10:SEG BB1*, (BB)
11:IN HPLNLM
12:SEG BB2*, (BB)
13:IN PPLNLM
14:END

```

(A) CALCOMP.

```

1:LIB *SD4060, MSC*LOCALIB, MSC*PLTLIB
2:LIB MSC*LOCALIB, MSC*PLTLIB, *SD4060
3:NOT IMAGE
4:SEG MAIN
5:IN PLOTTR
6:SEG AA*
7:IN COMPLT
8:SEG BB*, (MAIN)
9:IN THRODS
10:SEG BB1*, (BB)
11:IN HPLNLM
12:SEG BB2*, (BB)
13:IN PPLNLM
14:END

```

(B) SD4060.

```

1:LIB *HAZEL.
2:NOT IMAGE
3:NOT HAZEL.MODE2
4:NOT HAZEL.MAIN
5:SEG MAIN
6:IN PLOTTR
7:SEG AA*
8:IN COMPLT
9:SEG BB*, (MAIN)
10:IN THRODS
11:SEG BB1*, (BB)
12:IN HPLNLM
13:SEG BB2*, (BB)
14:IN PPLNLM
15:END

```

(C) MOPS.

FIGURE 34 MAP ELEMENTS FOR PLOTTR.

RDPRO: POST PROFILE TAPE READ PROGRAM

RDPRO is a small computer code designed to selectively read and dump tapes that are produced by the large and complex POST program. POST is a generalized point mass targeting and optimization program for powered or unpowered vehicles. The operation of POST is described in reference 32. The nominal method of operation of POST is to produce a printed report, a subset of which is placed upon an output for the processing by RDPRO.

Upon execution RDPRO expects from the card reader (terminal if demand) a set of directives in namelist form. It then reads the POST output tape and selectively extracts the desired parameters which are then placed upon a temporary secondary unit for introduction into an EDIN technology data base. They may also be listed if so desired.

Program Usage

Having established a profile tape using the POST program, a user can call on the RDPRO program to reformat the data to that of the EDIN data base. Shown below is a list of input parameters which are read in the INPRT namelist data.

PROFIL	Logical unit number of the POST output tape. Default is 1.
AAMURS	An array of hollerith names that are the desired variables to be extracted from the PROFIL tape. The default list is TIME, ALTITO, VELR, GAMMAR, PITI, WPROP, THRUST, ASMG, INCPCH and DYNP.
NUMVRS	Number of named variables being extracted. Default is clearly 10.
IPRO	The print flag. Equal to zero produces a listing, non zero - no listing. Default is non-zero.
CTTM	A two (2) word array defining the bounding times for the extracted data. Default is 0., 1.E10.
NMLIST	The unit number for temporary storage of the extracted information.
MOVE	The number of cases to skip before beginning an extraction run. POST cases may be stacked on an output tape.

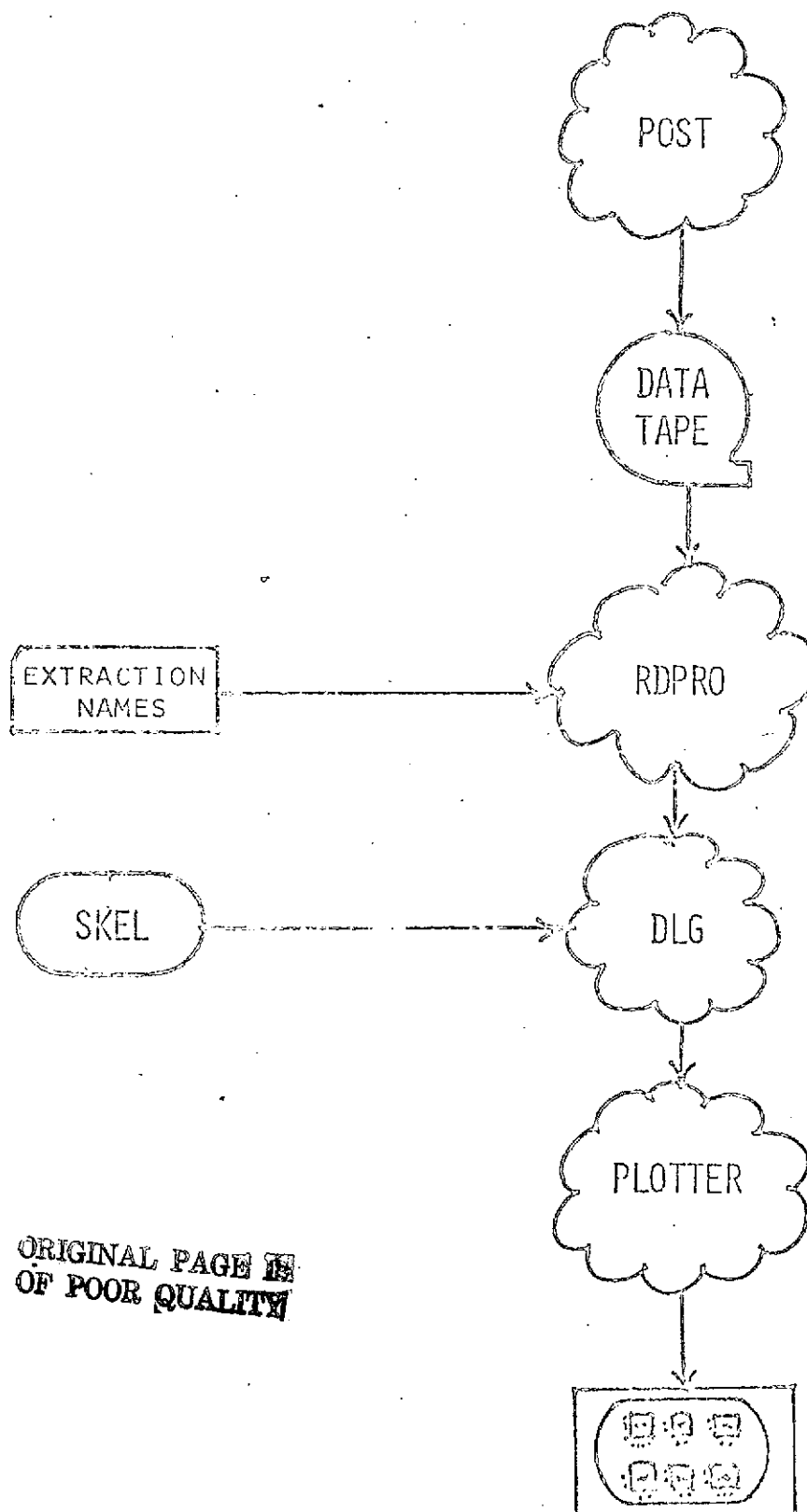
Example Input. -

```
$INPRT  NUMVRS=3,  CTIME=0.,100.  $
```

This extracts only time, altitude and velocity for the first 100 seconds of the launch sequence.

Program Output

Figure 35 is a logic/data flow of how RDPRO was used to produce the six plots seen in figure 36 . These data are a real life case study of launch of the S0147B Shuttle using J2S engines.



ORIGINAL PAGE 12
OF POOR QUALITY

FIGURE 35 USE OF RDPRO TO GENERATE TRAJECTORY PLOTS

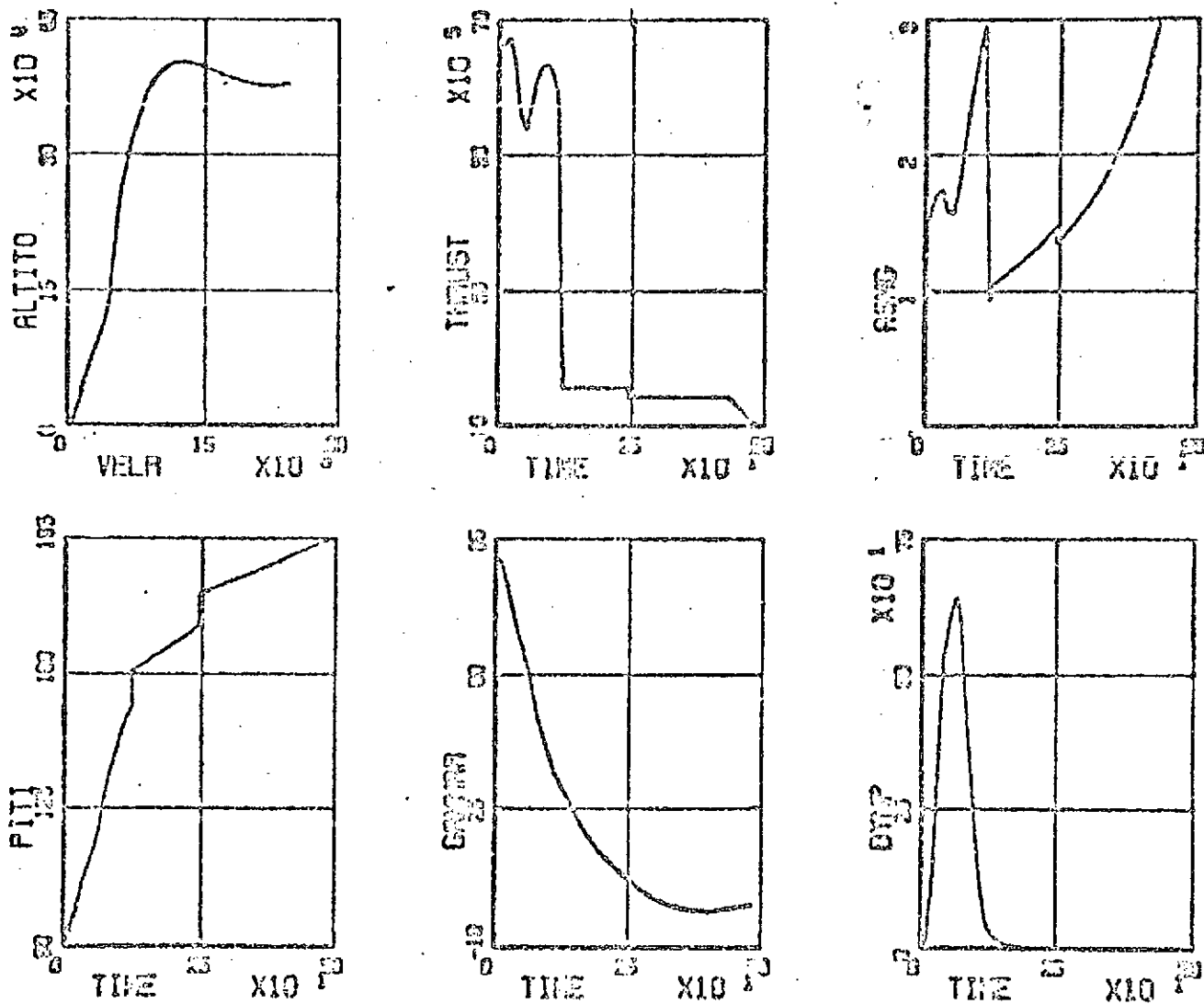


FIGURE 36 ILLUSTRATION OF TRAJECTORY PLOTS

ORIGINAL PAGE IS
OF POOR QUALITY

SFIT: A SURFACE FITTING PROGRAM

The SFIT program defines surfaces for objects that have a plane of symmetry. It was developed for the purpose of generating and/or modifying fuselage type surfaces. A parametric cubic patch method is used to define the surface over a set of cross sections that the user in the form of corner point geometry.

Program Description

The SFIT program can be used to generate new fuselage type surfaces or to modify existing surfaces. The surfaces must be symmetric about the X-Z plane with the X axis positive in the aft to forward direction and the Z axis positive in the bottom to top direction. The user defines the surface shape by means of cross section definitions. If the surface is to be a modification of existing geometry, the station at which the modification is to begin and the station at which the modification is to end are input to SFIT. The program then interpolates the existing surface geometry data and defines the cross sections at the ends of the segment to be modified. The user, whether describing new surfaces or modifying old geometry, inputs a series of cross section definitions in a fore to aft sequence. Because of the symmetry restriction, only one half of the cross section needs to be described. The description must include the X axis station of the cross section, a set of points (up to 40) which lie on the cross section and the number of points in the set. The points are defined by their Y,Z coordinates and are input in a bottom to top order. Figure 37 depicts the coordinate system and some cross section definitions.

The SFIT program uses a version of the Coons Bi-Cubic Parametric Patch Method to generate the surface. This method requires that the area to be fitted be bounded by four curves as shown in figure 38. SFIT uses the cross section definitions and a combination of spline fitting and LaGrange interpolation methods to generate the parametric boundary curves.

In order to insure a good surface fit, the area between each cross section is divided into nine sections, or patches. Each patch requires a set of boundary curves as mentioned above. The boundary curves labeled U1 and U2 in figure 38 can be conveniently constructed from the user supplied cross section definitions. First each cross section definition is expanded to ninety points by spline fitting for intermediate points. The total length of the cross section definition is computed and the ten breakpoints required to divide the cross section into nine equal length

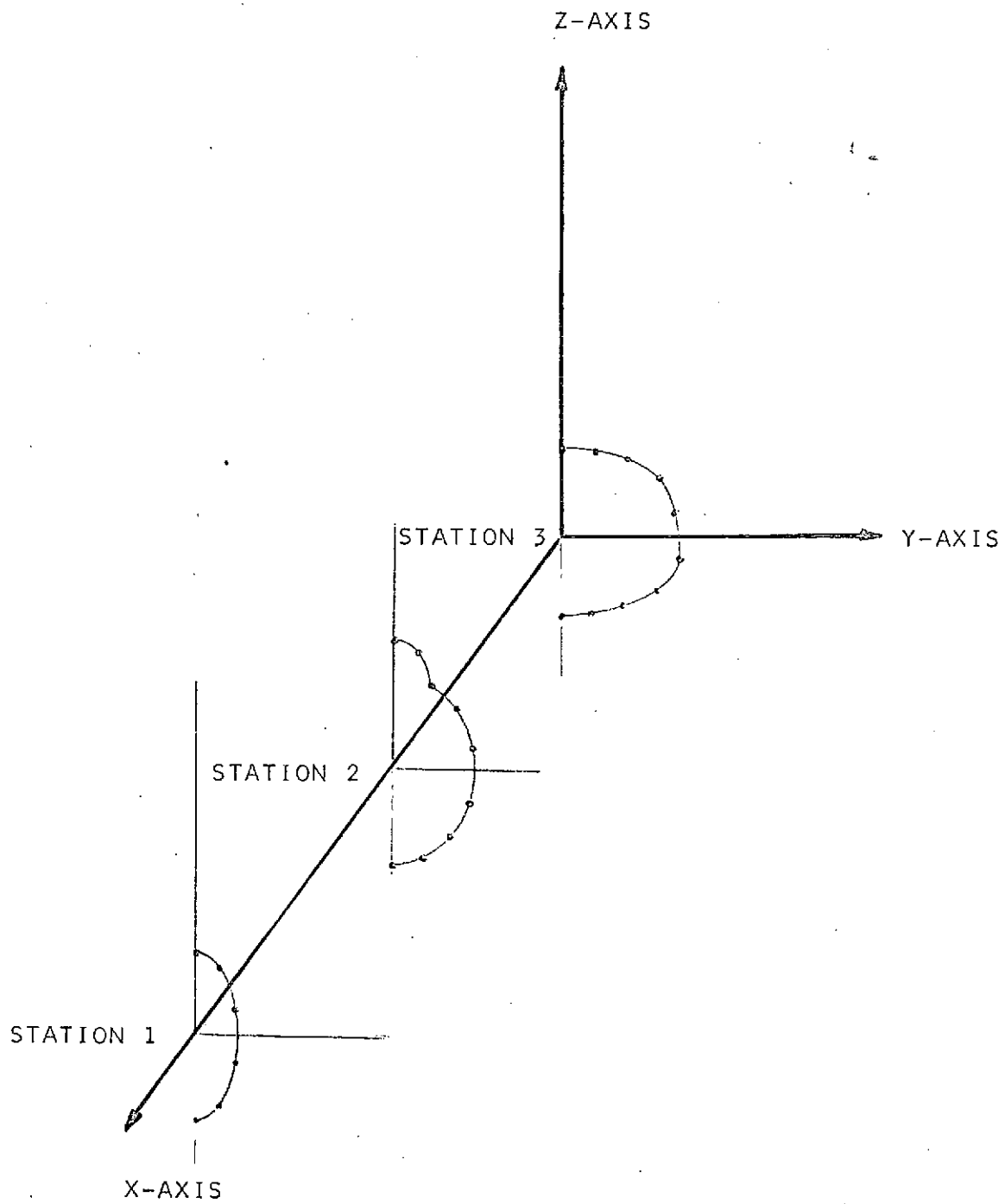


FIGURE 37 SFIT COORDINATE SYSTEM DEFINITION.

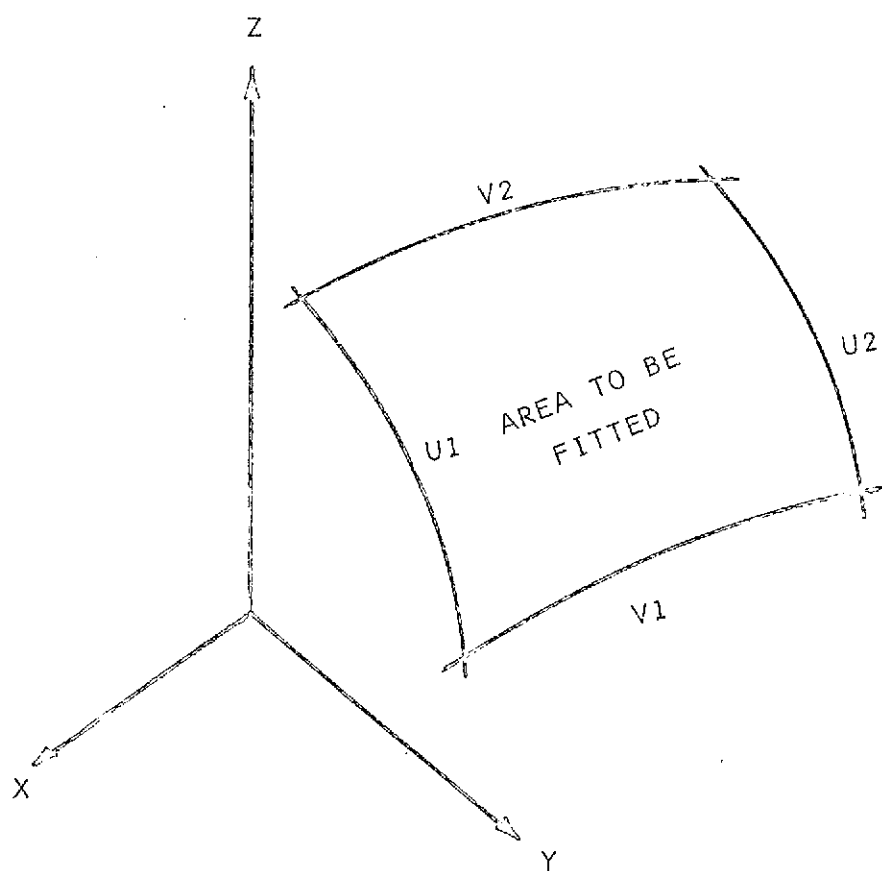


FIGURE 38 SURFACE BOUNDARY CURVES.

segments are computed. These line segments become the U boundary curves for the nine surface patches between cross sections. For example, segment one from cross section one and segment one from cross section two are used as U1 and U2 in the first surface patch between the two sections.

After each cross section has been divided into nine equal length segments and the segment breakpoints for each cross section recorded, the V curves can be generated. Each V curve is defined by fitting a curve through a group of data points. This group of points consists of one segment breakpoint from each cross section. For example, the curve labeled V1 for the first patch in a section is defined by a curve fitted through the first segment breakpoint from every cross section; the curve labeled V2 for the first section fits the set of points consisting of the second segment breakpoints from every cross section. Now that the U and V boundary curves are defined, the surface can be fitted by the parametric cubic method.

The SFIT output is in the form of cornerpoint geometry with status indicators to indicate start of section, start of component and end of component. The output geometry can be used for display and analysis just as Gentry geometry is used. Figures 39 and 40 are examples of SFIT modified geometry and new geometry.

Program Usage

Control Cards. - The program case data is proceeded by the control cards required to retrieve the WAB program from permanent storage and execute the program. The control cards needed to use SFIT at Johnson Spacecraft Center are:

```
@RUN RUNID,ACCOUNT,ORGANIZATION
@QUAL EX42-00002
@USE 8,OUTPUT FILE
@XQT *WAATS2.OFIT
    { SFIT DATA }
@FIN
```

Program Input

The SFIT program accepts three types of input data:

1. Geometry Modification Data in NAMELIST Format (\$IN).
2. Cross Section Data in NAMELIST Format (\$SECT).



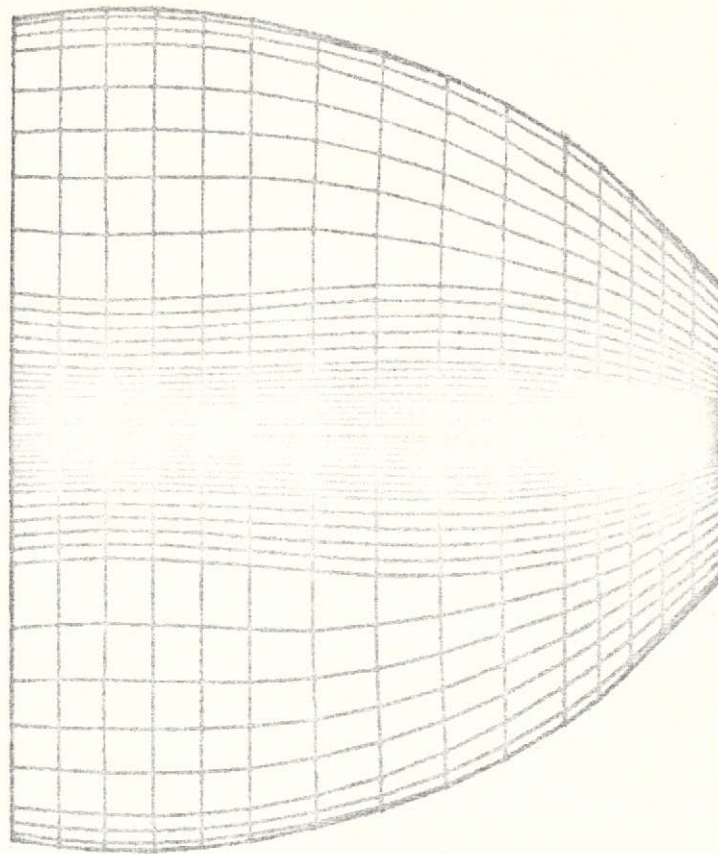
S0147B FUSELAGE



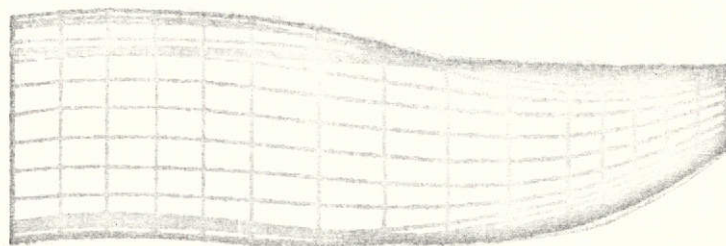
MODIFIED S0147B FUSELAGE

**ORIGINAL PAGE IS
OF POOR QUALITY**

FIGURE 39 GEOMETRY MODIFICATION WITH SFIT.



PLAN VIEW



SIDE VIEW

ORIGINAL PAGE IS
OF POOR QUALITY

FIGURE 40 SURFACE FROM ARBITRARY SECTIONS.

3. Cornerpoint Geometry in Fixed Format.

These input sets may be combined in various ways depending upon the input parameters in \$IN. The input sets are described below.

\$IN Inputs. -

<u>NAME</u>	<u>DESCRIPTION</u>	<u>DEFAULT</u>
FLAG	Geometry Modification Flag =1, Existing Geometry is to be Modified. =0, New Geometry is to be Created.	0.
START	Station at which Region of Geometry Modification is to begin.	0.
END	Station at which Region of Geometry Modification is to End.	0.

\$SECT Inputs. -

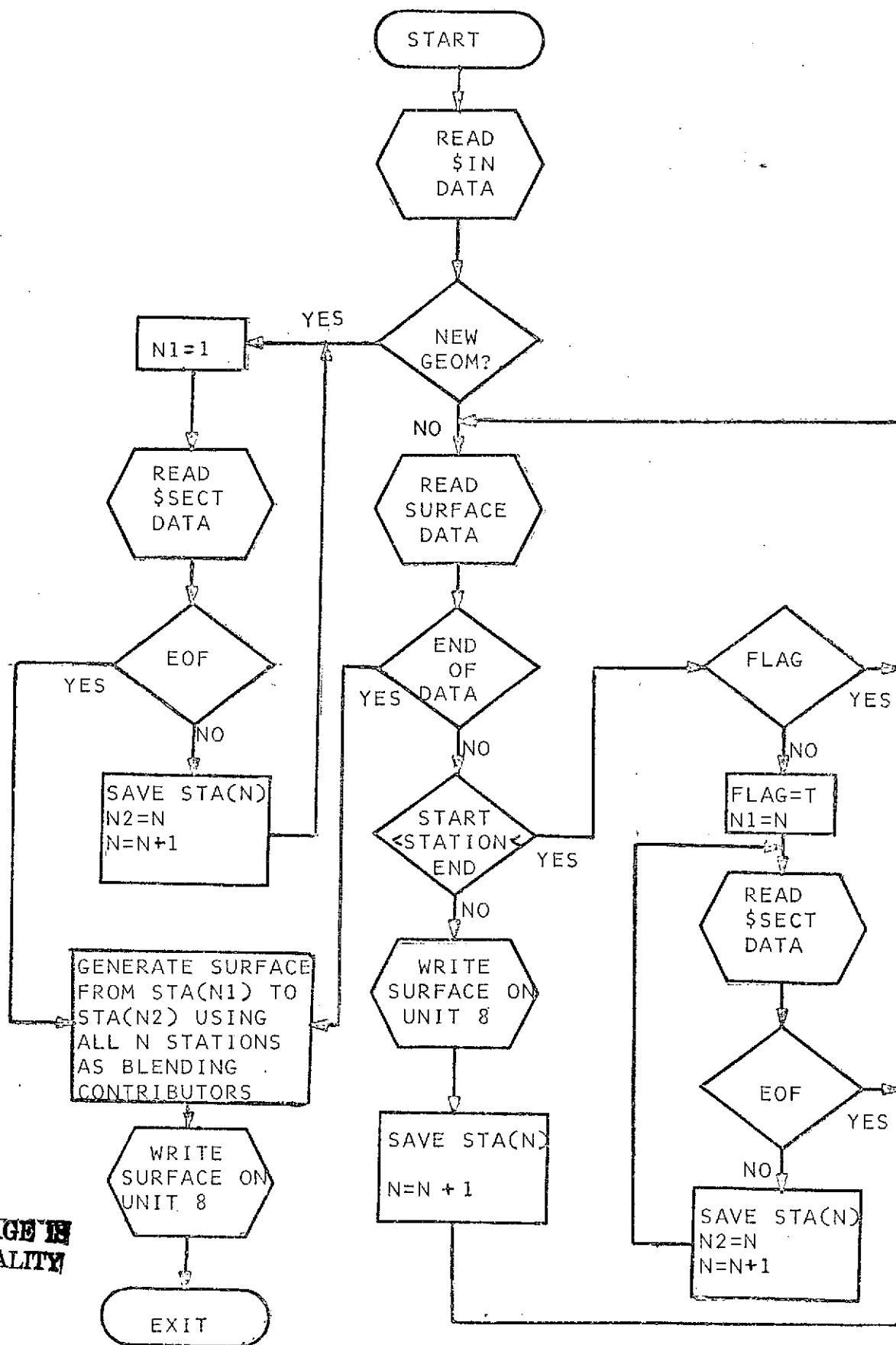
<u>NAME</u>	<u>DESCRIPTION</u>	<u>DEFAULT</u>
STA	Station at which Cross Section is being Defined.	0.
NPTS	Number of Points in Cross Section Definition.	0.
Y	Array of Y Coordinates of the 'NPTS' Points.	0.
Z	Array of Z Coordinates of the 'NPTS' Points.	0.

Program Flow Logic

Figure 41 gives an overview of the SFIT program flow.

Program Output

The program output consists of cornerpoint geometry. This geometry is defined by a set of points in three dimensional space. The sets of X,Y,Z points are the cornerpoints of quadrilateral elements which define the surface. An additional parameter is attached to each set of coordinates. This parameter is a status flag which indicates if the point is the first point of a component, first point of a section, last point of a component or none of these. The output goes to data set eight (8) which may be user assigned.



ORIGINAL PAGE 11
OF POOR QUALITY

FIGURE 41

SIMPLIFIED FLOWCHART OF SFIT PROGRAM.

SIZER: A PRELIMINARY SIZING PROGRAM FOR
LAUNCH VEHICLES.

The SIZER program performs preliminary sizing for a launch vehicle composed of up to ten stages. User inputs of payload, specific impulse of each stage, ideal velocity of each stage and mass fraction of each stage are required. Program results are based on the ideal velocity equation,

$$V = g \cdot I_{sp} \cdot \ln(MR)_c$$

where MR is the ratio of the mass of the vehicle at the start of the stage to the mass of the vehicle at the end of the stage, I_{sp} is the specific impulse and g is gravitational acceleration. It should be restated that SIZER gives a preliminary sizing estimation based upon ideal velocity relationships; no drag losses or gravity losses are included. SIZER outputs a summary which presents the sizing information for each stage.

Program Description

The SIZER program uses the ideal velocity equation

$$V = g \cdot I_{sp} \cdot \ln(MR)$$

to perform preliminary sizing of an n stage launch vehicle. SIZER requires the user to define the following inputs:

n	Number of Stages.
W_{pay}	Payload of Vehicle
I_{sp}	Specific Impulse of each Stage.
V	Ideal Velocity of each Stage.
λ'	Mass Fraction of Each Stage.

These inputs, when applied to the ideal velocity equation, determine the total weight of each stage of the boost vehicle. The vehicle is sized from the top down so that the payload of a stage is the sum of the weights of every stage above it.

The stage weight solution can be determined as follows:

$$MR = \frac{W_s}{W_e}$$

where W_s is the weight at the start of the stage and W_e is the weight at the end of the stage. These weights can be expressed as:

$$W_s = W_{pay} + W_{inert} + W_{prop}$$

and

$$W_e = W_{pay} + W_{inert}.$$

where W_{inert} is the inert weight of the stage and W_{prop} is the stage propellant weight. Now, the mass fraction, λ' , is defined to be:

$$\lambda' = \frac{W_{prop}}{W_{prop} + W_{inert}}$$

so that:

$$W_{inert} = \frac{W_{prop}}{\lambda'} - W_{prop} = W_{prop} \frac{(1 - \lambda')}{\lambda'}$$

The start weight and end weight can then be expressed as:

$$W_e = W_{pay} + \frac{W_{prop}}{\lambda'}$$

and

$$W_e = W_{pay} + \frac{W_{prop} (1 - \lambda')}{\lambda'}$$

The mass ratio then becomes:

$$MR = \frac{\lambda' W_{pay} + W_{prop}}{\lambda' W_{pay} + W_{prop} (1 - \lambda')}$$

The ideal velocity equation can be written as:

$$\ln(MR) = \frac{V}{g I_{sp}} = K$$

or

$$MR = e^K.$$

After substitution, it can be seen that:

$$W_{prop} = \frac{\lambda' W_{pay} (e^K - 1)}{1 - e^K (1 - \lambda')}$$

The inert weight, W_{inert} , can now be computed from the mass fraction equation. The total stage weight is the sum of the inert and propellant weights, and the stage sizing is complete.

This process is repeated for each of the n stages and both printed output and EDIN data base output is provided. This output will be discussed in the following section.

Program Usage

Control Cards. - The program case data is preceded by the control cards required to retrieve the SIZER program from permanent storage and execute the program. The control cards needed to use SIZER at Johnson Spacecraft Center are:

```
@RUN RUNID,ACCOUNT,ORGANIZATION
@QUAL EX42-00002
@XQT *WAATS2.OSIZER
      SIZER DATA
@FIN
```

Input Procedure

SIZER accepts input data in NAMELIST format (\$IN). Data names, descriptions and default values are defined as follows:

<u>NAME</u>	<u>DESCRIPTION</u>	<u>DEFAULT</u>
PAYLD	Vehicle Deliverable Payload - Lbs.	150000.
NSTAGE	Number of Stages -- 10 Max.	2
VIDEAL	Array of Ideal Velocities - 1 per Stage	8380., 21293.,
XLAMDP	Array of Mass Fractions - 1 per Stage	.9, .9,
XISP	Array of Specific Impulses -- 1 per Stage	296., 455.,
WPEST	Array of Estimated Propellant - 1 per Stage	2.E6, 1.E6
CFACTR	Convergence Factor	1.
LU	Data Set Number for EDIN Output	14.

Flow Logic

Figure 42 shows the flow logic for the SIZER program.

Program Output

The SIZER program outputs two types of output. The first type of output consists of a sizing report which represents the sizing parameters broken out for each stage. Figure 43 shows an example of this output for a two stage vehicle. The second

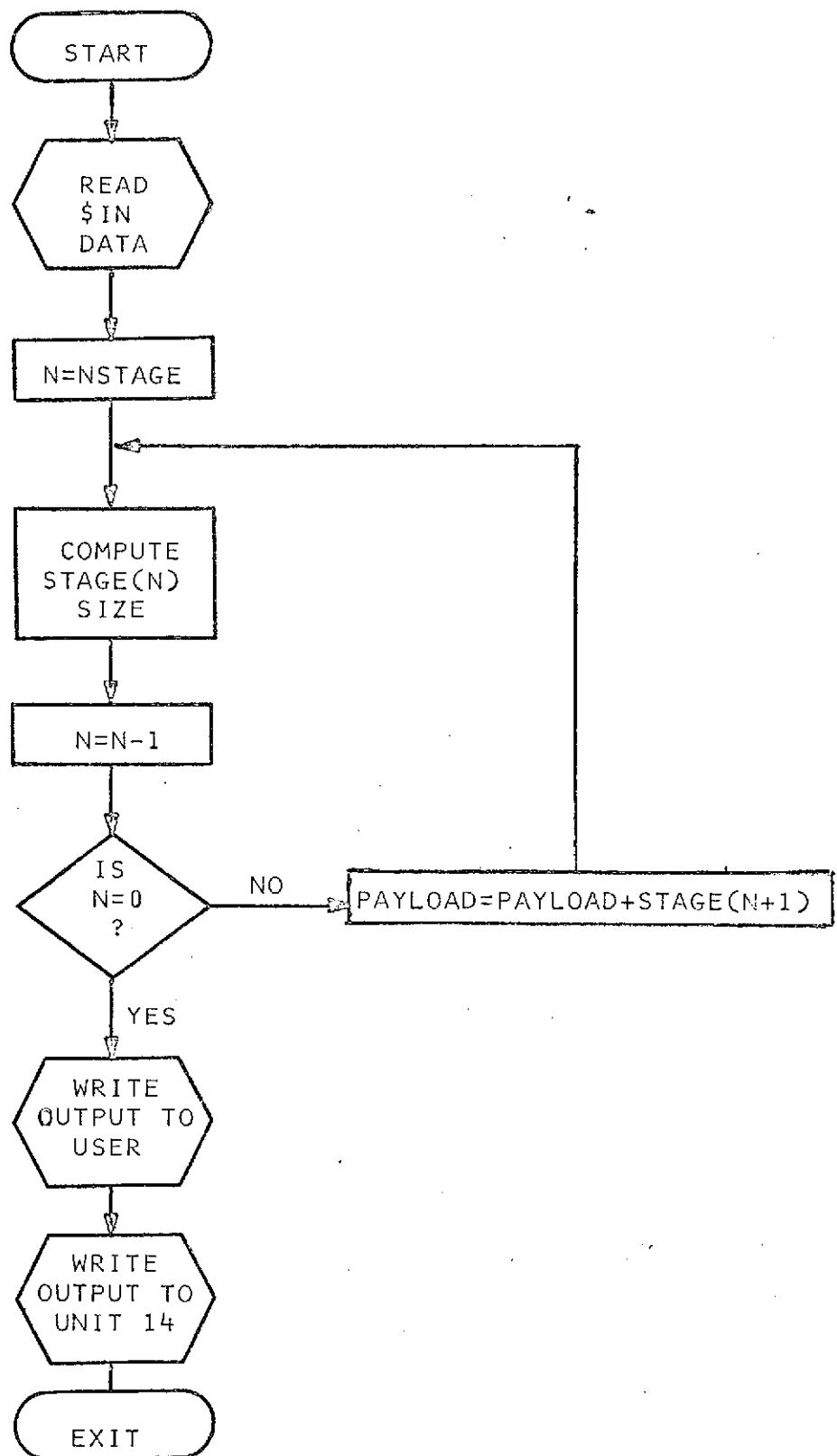


FIGURE 42 SIMPLIFIED FLOWCHART OF SIZER PROGRAM.

C T R F 2
 PWLDR= 20990.0

INERT WEIGHT		15046.
NOSS PLATE	935	
NOSS RATIO	3.884	
NOSS RATIO	206	
BURNOUT WEIGHT		75046.
PROPELLANT WEIGHT		216223.
VELOCITY	19866.	
SPECIFIC IMPULSE	205.3	
GROSS STAGE WEIGHT		291488.

C T R F 1
 PWLDR= 20168.0

INERT WEIGHT		38781.
NOSS PLATE	935	
NOSS RATIO	2.009	
NOSS RATIO	226	
BURNOUT WEIGHT		330263.
PROPELLANT WEIGHT		537283.
VELOCITY	10152.	
SPECIFIC IMPULSE	218.0	
GROSS STAGE WEIGHT		868115.

FIGURE 43 SIZER OUTPUT.

type of output consists of the computed sizing parameters being added to the EDIN data base. The output is sent to the data set specified by the variable LU in the \$IN namelist input. This data includes propellant weight, inert weight, start burn weight, end burn weight and payload of each stage. The names WPR, WI, WSB, WEB and WPL are attached to the output to the EDIN data base.

TANK: A PRELIMINARY TANK DESIGN PROGRAM.

The TANK Program is a highly interactive technology module designed for use in a preliminary tank design. The program will generate combined oxidizer and fuel tank design characteristics from input specifications. The program can operate in the batch or demand mode. In the demand mode the program requests data elements by name and short definition. In addition, an option to generate cornerpoint geometry in Gentry format, reference 27, is provided for image viewing and geometry manipulation. User inputs to the program include design criteria such as an O.F. factor, total propellant weight, stress factors, tank thicknesses, etc.

Program Description

The TANK Program utilizes standard engineering algorithms for the calculation of design characteristics. Standard engineering nomenclature and units for the derivation and output of data are used. Program organization is illustrated by the simplified flowchart in figure 44.

The required tank volumes are computed as a function of propellant ratios and ullage requirements. Standard hoop stress equations are used to compute tank thickness estimates. The elliptical cap thickness is determined as a percentage of the cylindrical thickness. Tank weights are based on the specific weights of the material chosen for the tank. Total weight estimates are based on accessory estimates, tank weights and propellant weights. Tank lengths are computed as a function of the total volume requirements.

An option to generate cornerpoint geometry in Gentry format is provided for image viewing. If selected, the program will read all previously generated data in addition to user inputs for intertank separation and the number of X-stations per tank cap. These data are used to generate the tank geometry. Once complete, the geometry is available on Unit 8.

Physical Characteristics

The physical characteristics of the TANK Program are summarized as follows:

HOST COMPUTER:	UNIVAC EXEC 8 (1110)
PROGRAM FILES:	EX42-00002*WAATS2. (SOURCE, RELOCA - TABLES, ABSOLUTE)
	EX42-00002*WAATS2. (MAP ELEMENT)

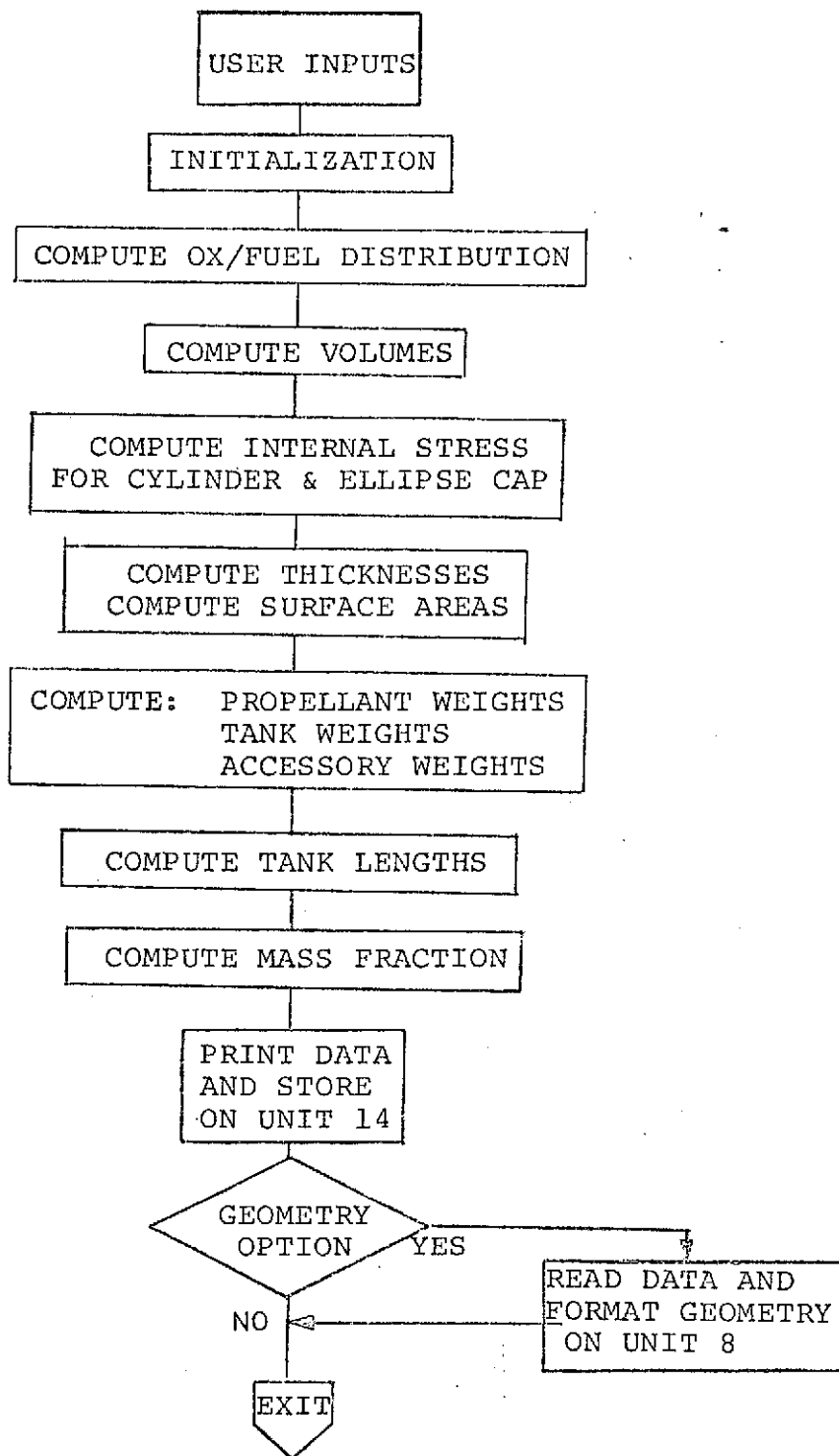


FIGURE 44 SIMPLIFIED TANK PROGRAM FLOW CHART.

ABSOLUTE ELEMENT: OTANK
 LANGUAGE: FORTRAN V
 PROGRAM SIZE: 6950 DECIMAL
 CARD SOURCE: +425

Program Usage

The computer program usage requirements described in this section are generally orientated toward the Exec 8 1110 version and specifically towards the Johnson Spacecraft Center installation. The actual program input requirements described are applicable wherever the program is installed but the control cards for the retrieval and execution of the program will differ from computer to computer. A typical runstream for TANK is illustrated below:

	<u>ID</u>	<u>ACCOUNT NUMBER</u>	<u>ORGANIZATION</u>
@RUN	AB123,	1230C-IO88-C,	EX42-00002
@ASG,T	14.	Assign Output File 14	
@ASG,T	8.	Assign Geometry Output File 8.	
@XQT	EX42-00002*WAATS2.OTANK		
	data input		
	data input		
	.		
	.		
	.		
	.		
@FIN	Termination		

Program Input. - The following is a listing of the required inputs for a TANK program execution. Since the inputs are made in response to a program inquiry, only the inquiries are shown.

ENTER AN O.F. RATIO	(Computer)
_____ (decimal entry)	(User)
ENTER 1 IF LOX/LH SYSTEM, ENTER 2 IS OTHER	(Computer)
_____ (integer entry)	(User)
ENTER YOUR TOTAL PROPELLANT REQUIREMENTS - LBS.	(Computer)
_____ (decimal entry)	(User)

IF A 2 IS ENTERED THE PROGRAM WILL ADDRESS THE USER AS FOLLOWS:

ENTER SPECIFIC WEIGHT OF OXIDIZER - LBS/CU IN	(Computer)
_____(decimal entry)	(User)
ENTER SPECIFIC WEIGHT OF FUEL - LBS/CU IN	(Computer)
_____(decimal entry)	(User)
ENTER % ULLAGE OXIDIZER	(Computer)
_____(decimal entry)	(User)
ENTER % ULLAGE FUEL	(Computer)
_____(decimal entry)	(User)
ENTER MAX OXIDIZER PRESSURIZATION	(Computer)
_____(decimal entry)	(User)
ENTER MAX FUEL PRESSURIZATION	(Computer)
_____(decimal entry)	(User)
ENTER MAX TANK DIAMETER	(Computer)
_____(decimal entry)	(User)
ENTER AN OXIDIZER TANK STRESS S.F.	(Computer)
_____(decimal entry)	(User)
ENTER A FUEL TANK STRESS S.F.	(Computer)
_____(decimal entry)	(User)
ENTER TYPE ELLIPSE CAP-DEG	(Computer)
_____(decimal entry)	(User)
OX TANK MATERIAL: ENTER 1 FOR AL, 2 FOR TI,	
_____3 FOR OTHER	(Computer)
_____(integer entry)	(User)
FL TANK MATERIAL: ENTER 1 for AL, 2 FOR TI,	
_____3 FOR OTHER	(Computer)
_____(integer entry)	(User)

IF A 3 IS ENTERED THE PROGRAM WILL SOLICIT INFORMATION FROM USER AS FOLLOWS:

ENTER OXIDIZER TANK MATERIAL SPECIFIC WEIGHT -	
_____LBS/CU IN	(Computer)
_____(decimal entry)	(User)
ENTER FUEL TANK MATERIAL SPECIFIC WEIGHT - LBS/CU IN	(Computer)
_____(decimal entry)	(User)
ENTER OXIDIZER TANK MATERIAL ULT STRESS - LBS	(Computer)
_____(decimal entry)	(User)
ENTER FL TANK MATERIAL ULT STRESS - LBS	(Computer)
_____(decimal entry)	(User)
ENTER % EST WEIGHT OF OXIDIZER TK ACCESSORIES	(Computer)
_____(decimal entry)	(User)
ENTER % EST WEIGHT OF FUEL TK ACCESSORIES	(Computer)
_____(decimal entry)	(User)
ENTER 1 TO GENERATE DESIGN POINTS	(Computer)
_____(integer entry)	(User)
ENTER NO OF X STATIONS PER CAP	(Computer)
_____(decimal entry)	(User)
ENTER TANK SEPARATION	(Computer)
_____(decimal entry)	(User)

Program Output. - In addition to the formatted geometry which is output by TANK, the following data is output to the printer and to Unit 14.

O.F. RATIO

TYPE OF OXIDIZER

TYPE OF FUEL

SPECIFIC WEIGHT OF THE OXIDIZER

SPECIFIC WEIGHT OF THE FUEL

OXIDIZER TANK MATERIAL

The EDIN data base output is written on Unit 14 and contains the following information:

SA	An array of two elements containing the surface areas for the oxidizer and fuel tanks.
LT	Array of two elements containing the lengths of the oxidizer and fuel tanks.
VOL	Array of two elements containing the total volume of the oxidizer and fuel tanks.
WP	Array of two elements containing the propellant weights of the oxidizer and fuel tanks.
WTK	Array of two elements containing the weights of the oxidizer and fuel tanks.
WAC	Array of two elements containing the accessory weights of the oxidizer and fuel weights.
WTOT	Array of two elements containing the total weight of the oxidizer and fuel tanks.

VL70: A PROGRAM FOR READING AERODYNAMIC
DATA TAPES.

VL70 is a general purpose utility program designed to extract and reformat specific incremental aerodynamic information from standard aerodynamic data tapes. Input to the program is namelist and the program can be executed in batch or demand. Output of the program consists of formatted data which can be easily interfaced to existing EDIN application software.

Program Description

The basic structure of the VL70 program is illustrated by figure 45 . The program is drawn by a schedule which provides the following data:

- Mach Number.
- Angle of Attack (nominal, upper and lower).
- Limit Body Flap Deflections.
- Limit Elevon Deflections.
- Limit Rudder Deflections.
- Data File Numbers.
- Independent Variable Names.
- Dependent Variable Names.

The schedule is usually divided into three parts; (1) the speed brake deflection schedule, (2) the body flap deflection and (3) the elevon deflection schedule. Each part reflects the Mach number, the angle of attack range, control deflections and file number.

The schedule of deflections is described to the computer by a series of 5 x 14 arrays (five deflection conditions and 14 Mach numbers). The deflection conditions represent speed brake deflections, elevon deflections and body flap deflections.

In addition, the alpha schedule is described as three (3) deflections at each of the fourteen (14) Mach numbers. The three (3) deflections represent the nominal, a high alpha and a low alpha.

For each deflection condition there is also a schedule of file numbers and independent variable names. These names, which ultimately become EDIN data base names, are not presented herein but may be obtained from the program listing.

The nominal schedule may be overridden by input in the \$INPT namelist as described by the section on program usage.

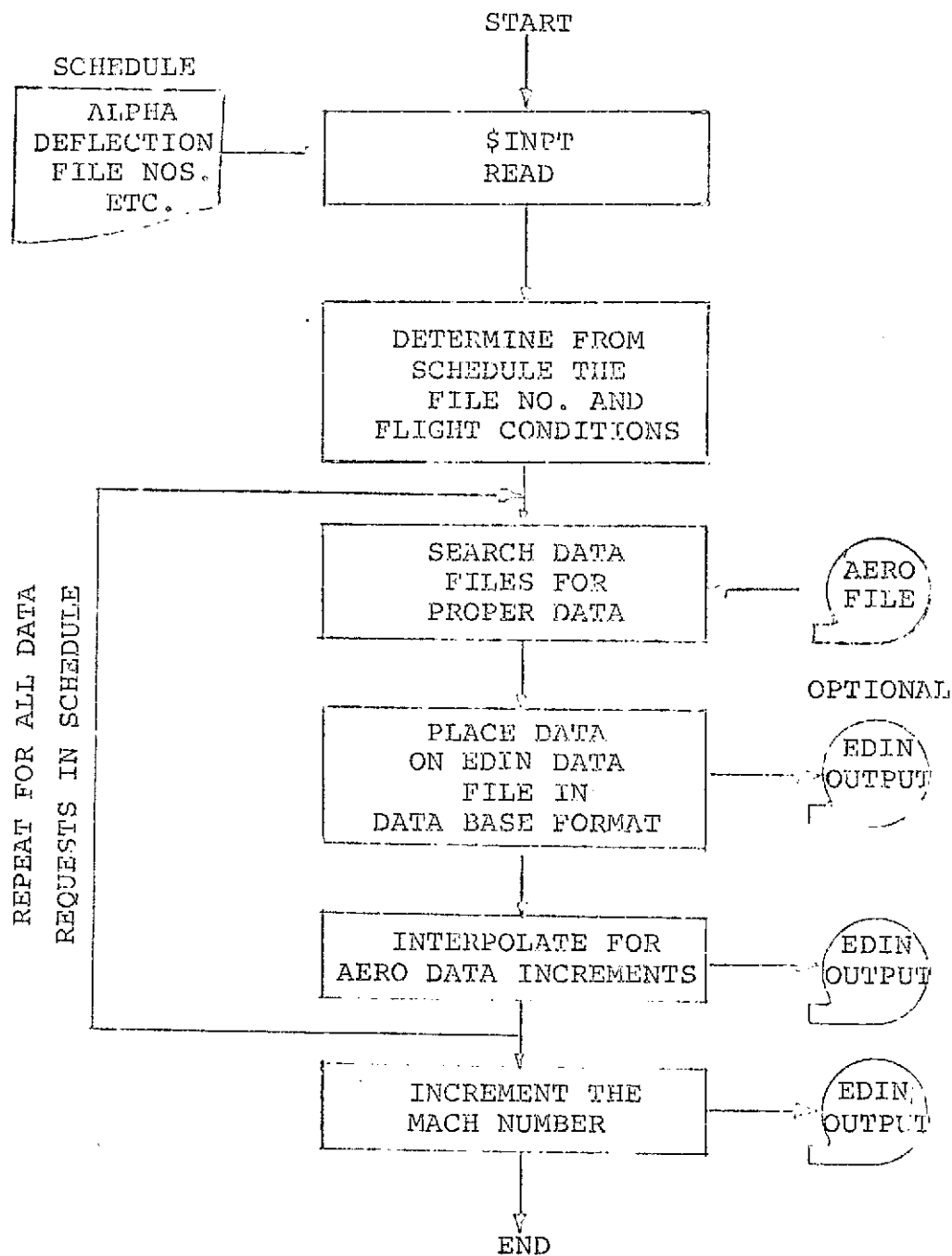


FIGURE 45 VL70 PROGRAM STRUCTURE

Physical Characteristics

HOST COMPUTER: UNIVAC EXEC 8 (1110)
PROGRAM FILES: EX42-00002*ODIN-ROMER.ANALY1 (Relocatable)
EX42-00002*ODIN-ROMER.ANALY1 (Source)
ABSOLUTE ELEMENT: ANALY1A
LANGUAGE: FORTRAN V
PROGRAM SIZE: 10000 DECIMAL
CARD SOURCE: + 1257

Program Usage

The computer usage requirements described in this section are generally orientated toward the Exec 8 1110 computer and specifically towards the Johnson Spacecraft Center installation. The actual program input requirements described are applicable wherever the program is installed by the control cards for retrieval and execution of the program will differ from computer to computer. A typical run stream for VL70 is illustrated below:

<u>ID</u>	<u>ACCOUNT NUMBER</u>	<u>ORGANIZATION</u>
@RUN AB123,	1230C-I088C,	EX42-00002
@ASG,T 14.	Assign temporary Unit 14 for output.	
@XQT EX42-00002*ODIN-ROMER2-ANALY1A	Execute Core.	
\$INPT		
\$	Namelist Input.	
@FIN	Termination Card.	

Program Input. - The VL70 program input is namelist (\$INPT) and is defined by figure 46

Program Output. - The VL70 program uses temporary Unit 14 as an output file. Figure 47 illustrates a typical output from BASAERO. The output will consist of interpolated values of aerodynamic data based on input specifications for file numbers and named variables.

\$INPT NAMELIST

JMACH	Schedule counter used to identify the current position in the deflection schedule.
MFILE	Maximum number of files to search for Aero Data.
ALPHAS(3,14)	Nominal upper and lower Alpha schedule for 14 Mach numbers.
DEFLEC(5,14)	Schedule of deflections for 14 Mach numbers in schedule. 1 - Speed Brake Schedule. 2 and 3 - Fore and Aft Elevon Schedule. 4 and 5 - Fore and Aft Body Flap Schedule.
CMFILE(5,14)	Schedule of pitching moment file numbers for 14 Mach numbers as described above.
CLFILE(5,14)	Schedule of lift coefficient file numbers for 14 Mach numbers as described above.
CMXYZ(3,5,15)	Sequence of independent variables (same for CM and CL) used in the tabulation of aerodynamic data on the CM and CL files.
CNAMES(5,2)	Dependent variable names assigned to the interpolated data from the Aero File.

FIGURE 46 INPUT DESCRIPTIONS FOR VL70.

ORIGINAL PAGE IS
OF POOR QUALITY

```

$
SUL70  DCLCSF=-.00020000000, .00030000000, .00200000000,
$
SUL70  DCHS3F=.00720000000, .00620000000, .00440000015,
$
SUL70  DCLCLF=-.05479999931, -.06560000004, -.07500000011,
$
SUL70  DCHCLF=.03963999989, .051000000105, .063179999586,
$
SUL70  DCLCLR=.055499999835, .051599999849, .046099999852,
$
SUL70  DCHCLR=-.07330000014, -.06621999981, -.09811999984,
$
SUL70  DCLBFF=-.00949999999, -.01200000000, -.01449999999,
$
SUL70  DCHBFF=.015399999992, .016399999994, .021399999997,
$
SUL70  DCL2FA=.016399999995, .016999999999, .016999999990,
$
SUL70  DCHBFA=-.02314999998, -.02759999999, -.03210000000,
$
SUL70  WDLH   =20.000999928,

```

FIGURE 47 OUTPUT FROM VL70 PROGRAM.

WAB: A PROGRAM FOR COMPUTING WEIGHTS
AND BALANCES.

The WAB program computes the volume, area and mass properties of a structure. The structure can be defined as a surface form, a set of black box components or a combination of both. Surface form definition is in the form of corner point geometry. WAB outputs include a mass property summary and EDIN data base additions.

Program Description

The WAB program computes volume, surface area, frontal area, weight, center of gravity, moments of inertia and products of inertia for bodies whose surface shapes are composed of a number of quadrilateral elements. The cornerpoints of the quadrilateral elements are defined by ordered sets of X,Y,Z coordinates. Computation of mass properties requires that some characteristics of the surface material be known, such as surface thickness and material density or weight per unit of the surface area. Also, if the coordinate system that the mass properties are referenced to is different from the coordinate system that the surface cornerpoints are defined in, then the Euler angles for the coordinate system transformation must be specified. Basically, WAB computes all of the attributes listed above for each quadrilateral element, transforms these to the reference system and sums for all the elements to arrive at the total mass properties of the body.

The problem of computing the properties of the quadrilateral element is greatly simplified by splitting the quadrilateral into two triangular elements with a common side being one of the diagonals of the original quadrilateral. Formula for computing the area, center of gravity, moments of inertia and products of inertia of a triangular element of finite thickness are easily derived or may be found in most physics handbooks. The element's mass properties are computed in a temporary coordinate system which has its X axis colinear with the base of the triangle, its Z axis normal to the plane of the triangle and its Y axis completes the right handed triad with the system origin at a corner of the triangle. All that remains to be done is to rotate the inertia tensor into a coordinate frame that is parallel to the reference frame and then translate to the reference system origin by the parallel axis theorem. At this point the mass characteristics of the triangular element are defined with respect to the origin of the reference coordinate system. This process is repeated for the other triangular element that comprised the original quadrilateral and the properties are summed. Every quadrilateral element of the surface is processed and its properties summed in the same fashion. After all of the surface shape has been processed, the center

of gravity for the entire structure is computed and the moments and products of inertia of the body about its center of gravity are computed by the parallel axis theorem.

The volume enclosed by the body is computed by accumulating the volumes capped by each triangular element. This is done by projecting the area of each element onto the X-Z plane and multiplying this area times the distance of the element's centroid from the X-Z plane. If the outside normal of the element is away from the X-Z plane, the elemental volume is added to the sum; otherwise it is subtracted.

The WAB program also accepts "black box" inputs to the mass properties accumulations. Weight, center of gravity location and the inertia tensor may be input as black box characteristics. The program is capable of handling any number of components whether defined by surface form data or black box form data and the forms may be intermixed.

Program Usage

Control Cards. - The program case data is preceded by the control cards required to retrieve the WAB program from permanent storage and execute the program. The control cards needed to use WAB at Johnson Spacecraft Center are:

```
@RUN RUNID,ACCOUNT,ORGANIZATION
```

```
@QUAL EX42-00002
```

```
@XQT *WAATS2.OWAB
```

```
WAB DATA
```

```
@EOF
```

```
@FIN
```

Input Procedure

The WAB program accepts three types of input data:

1. General information data in NAMELIST format (\$IN).
2. Black box definition data in NAMELIST format (\$BOX).
3. Surface cornerpoint data in fixed format.

These input sets may be combined in various ways depending upon the input parameters in \$IN and \$BOX. This section describes the input data requirements.

\$IN Input Set. -

<u>NAME</u>	<u>DESCRIPTION</u>	<u>DEFAULT VALUE</u>
IREFL	Symmetric Geometry Indicator. = 1, Surface Shape or Black Box Has a Symmetrical Counterpart on Other Side of X-Z Plane. = 0, No Symmetrical Counterpart.	0
DELX	X Distance between Reference System and Input System.	0.
DELY	Y Distance between Reference System and Input System.	0.
DELZ	Z Distance between Reference System and Input System.	0.
PSI	Yaw Euler Angle for Input to Reference System Rotation.	0.
THETA	Pitch Euler Angle for Input to Reference System Rotation.	0.
PHI	Roll Euler Angle for Input to Reference System Rotation.	0.
BLKBX	Black Box Input Data Indicator = .TRUE., Input Data will Come from \$BOX = .FALSE., Input Data Will be Fixed Format Surface Form.	.FALSE.
IWRITE	EDIN Data Base Output Request =1, WAB Outputs Will be Added to EDIN Data Base. = 0, No Additions to Data Base.	0
FACL	Conversion Factor for Units of Length.	12.
FACI	Conversion Factor for Inertia Tensor	1.
SUB	Subtraction Flag. = .TRUE., WAB Properties of the Next Sur- face or Black Box will be Sub- tracted from the Accumulative Totals.	.FALSE.

<u>NAME</u>	<u>DESCRIPTION</u>	<u>DEFAULT VALUE</u>
	= .FALSE., WAB Properties of the Next Surface or Black Box Will be Added to the Accumulative Totals.	
RHO	Density of the Surface Material	1.
H	Thickness of the Surface Material	1.
NAME	A Six Character Hollerith Name that is Attached to the Array that is Output to the EDIN Data Base.	NONE

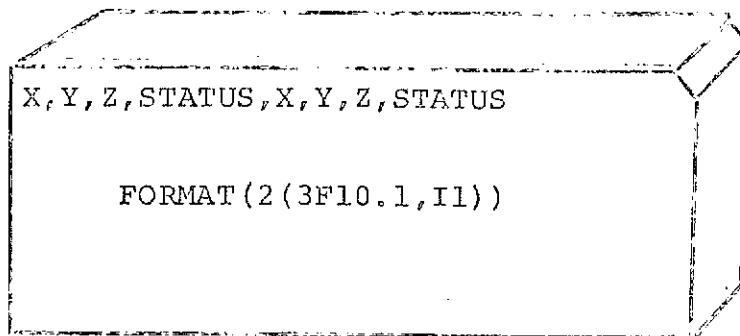
\$BOX INPUT SET. -

<u>NAME</u>	<u>DESCRIPTION</u>	<u>DEFAULT VALUE</u>
WT(1)	Weight of Black Box.	0.
WT(2)	X Center of Gravity.	0.
WT(3)	Y Center of Gravity.	0.
WT(4)	Z Center of Gravity.	0.
WT(5)	Moment of Inertia about X.	0.
WT(6)	Moment of Inertia about Y.	0.
WT(7)	Moment of Inertia about Z.	0.
WT(8)	XY Product of Inertia.	0.
WT(9)	XZ Product of Inertia.	0.
WT(10)	YZ Product of Inertia.	0.
WT(11)	Surface Area of Black Box.	0.
WT(12)	Volume of Black Box.	0.
WT(13)	Frontal Area of Black Box.	0.

Fixed Format Surface Data. - The surface cornerpoint data is read in two points at a time. The data includes X,Y,Z coordinates of each point and a status indicator for each point. The status values and their meanings are:

STATUS = 1; First Point of a Section.
STATUS = 2; First Point of a Component.
STATUS = 3; Last Point of a Component.
STATUS = 0; None of the above.

Each coordinate data field is specified to be F10.1 and each status indicator field is specified to be I1 with no blanks between fields. A surface data set is illustrated below:



Program Outputs

WAB outputs are of two types. One output is a mass properties report that is returned to the user. This data presents the characteristics of each surface form or black box that was input to WAB and the accumulative subtotals for all inputs. Figure 48 is an example of the WAB output. The other output is the optional output to the EDIN data base. This is essentially the same data that is presented in the mass properties report except that all of the data is packed into one array. The array is given a user specified name and then passed on to the EDIN data base.

Program Flow. - Figure 49 gives an overview of the WAB program flow.

WAB TANK

WT = 1744887.	YCG = .5731294-06	ZCG = 65.000005
XCG = -533.9599	IYY = .9867360+08	IZZ = .9867359+08
IXX = 8943624.	IXZ = -4.9115	IYZ = -.1402963-01
IXY = .1163292	UOL = 73381.63	AFR = 562.3780
ASF = 12116.71		

SUB 1

WT = 1744887.	YCG = .5731294-06	ZCG = 65.000005
XCG = -533.9599	IYY = .9867360+08	IZZ = .9867359+08
IXX = 8943624.	IXZ = -4.911511	IYZ = -.1402963-01
IXY = .1163292	UOL = 73381.63	AFR = 562.3780
ASF = 12116.71		

BODY

WT = 963576.5	YCG = .00000000	ZCG = 373.5639
XCG = -950.7733	IYY = .2661131+08	IZZ = .2661595+08
IXX = 3010939.	IXZ = -1.402411	IYZ = .00000000
IXY = .00000000	UOL = 30730.21	AFR = 389.4996
ASF = 6691.503		

SUB 2

WT = 2708383.	YCG = .5538360-06	ZCG = 174.7795
XCG = -685.4730	IYY = .1607649+09	IZZ = .1479121+09
IXX = .2471149+08	IXZ = -.1711870	IYZ = -.5658663-01
IXY = .2219290	UOL = 106161.8	AFR = 957.8776
ASF = 12803.22		

FIGURE 48 WAB OUTPUT.

ORIGINAL PAGE IS
OF POOR QUALITY

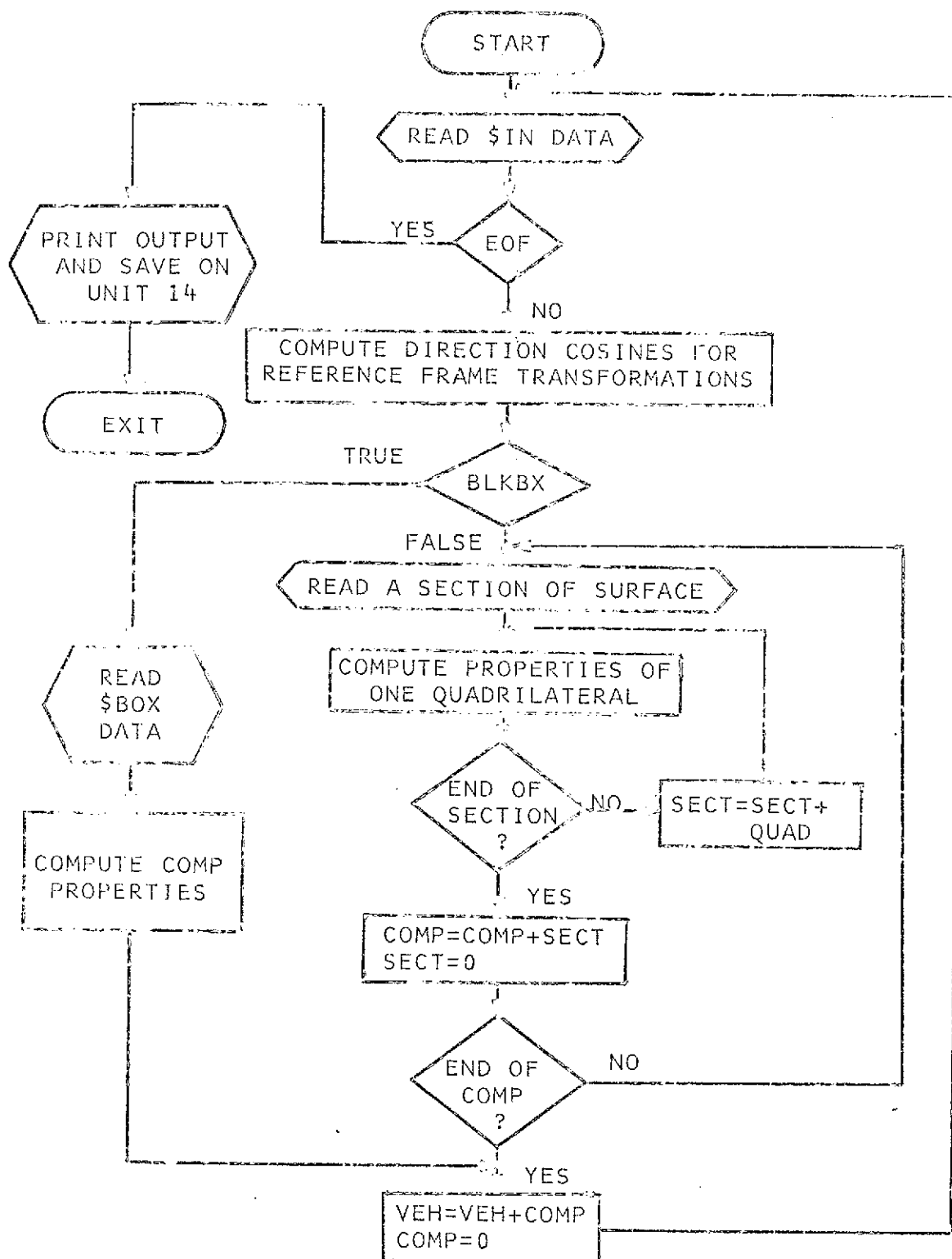


FIGURE 49 SIMPLIFIED FLOWCHART OF WAB PROGRAM.

APPLICATION OF THE SYSTEM

During the contract period several engineering investigations were conducted for the Engineering Analysis Division and Future Programs Office at the Johnson Spacecraft Center. These studies involved extensive use of the Engineering Design and Integration System (EDIN) and included investigation of several heavy lift booster concepts, advanced shuttle concepts studies, integrated space station study and others. These investigations involved the generation of geometry for digital display, aerodynamic separation studies, mass properties investigations, pressure/temperature studies, center of gravity analysis and performance evaluations including launch and trajectory analysis. As a result of these studies, the EDIN data base was expanded to its current level. Figure 50 summarized the current contents of the EDIN data base.

The following sections describe the particular study, analysis or evaluation which was performed.

Heavy Lift Booster (4 SRMS)

This study involved the generation of cornerpoint geometry for digital display. This particular booster was composed of a 4 SRM first stage, a single SRM booster second stage and a solid rocket third stage. The payload consisted of the shuttle orbiter (SO147B). As a result of this study, the geometry for this booster concept now resides in the EDIN geometry library.

Heavy Lift Booster (5 SRMS)

This study involved the generation of cornerpoint geometry for digital display. In addition, a mass properties evaluation was performed for the determination of volumetric requirements and center of gravity travel. This particular booster consisted of a 5 SRM first stage, a single SRM second stage and a solid rocket third stage. As a result of this study, the geometry and mass properties are readily available in the EDIN library.

Nuclear Waste Disposal Heavy Lift Booster

This evaluation involved the generation of cornerpoint geometry, a mass properties analysis and a trajectory analysis. This booster was conceived for disposal of nuclear waste in deep space. It consisted of a 5 SRM first stage, a single SRM second stage, a solid rocket third stage and liquid rocket engine configuration for the fourth, fifth and six stages. Once the geometry was created, separation studies involving shroud separation and staging conditions were duplicated using various EDIN

GEOMETRY:

SO147B SHUTTLE ORBITER.
SHUTTLE ORBITER EXTERNAL TANK.
SHUTTLE ORBITER SOLID ROCKET BOOSTERS.
ESRO SPACE STATION.
NAR SPACE STATION.
HEAVY LIFT BOOSTER (4 SRMS).
HEAVY LIFT BOOSTER (5 SRMS).
SO147B STRETCHED SHUTTLE ORBITER.
NUCLEAR WASTE DISPOSAL HEAVY LIFT BOOSTER.
SO147B HEAVY LIFT BOOSTER.
J2S ENGINE.
LR87 BOOSTER.
LR87 ENGINE.
SSME ENGINE.

AERODYNAMIC DATA:

SHUTTLE ORBITER BASELINE HYPERSONIC.
STANDARD AERODYNAMIC DATA TAPES.

MASS PROPERTIES DATA:

SO147B SHUTTLE ORBITER.
SO147B SHUTTLE ORBITER EXTERNAL TANK.
SO147B SHUTTLE ORBITER SOLID ROCKET BOOSTER.
HEAVY LIFT BOOSTER THIRD STAGE.

PERFORMANCE DATA:

SHUTTLE ORBITER W/SSME (NOMINAL).
SHUTTLE ORBITER W/4 X J2S.
SHUTTLE ORBITER W/5 X J2S.
SHUTTLE ORBITER W/SSME (AOS).
NUCLEAR WASTE DISPOSAL HEAVY LIFT BOOSTER.
SHUTTLE ORBITER SRM.

FIGURE 50 EDIN DATA BASE CONTENTS.

display devices. These separation evaluations were performed in conjunction with the trajectory analysis. The trajectory analysis included evaluations of performance to low earth orbit.

SO147B Payload/Body Volumetric Study

A volume and mass property analysis was conducted on the shuttle orbiter (SO147B) to determine the possibility for fuel storage. Volume evaluation on the fuselage including the current payload area were conducted. In addition, the wing root areas were evaluated. Compensation for equipment storage as well as on-board systems were considered.

SO147B Heavy Lift Booster

This study involved the investigation of a heavy lift booster concept consisting of twin orbiters, an external tank and a JATO pack for initial boost assistance. Two separate investigations were conducted to determine the possibility of using a 15 foot and a 16 foot diameter double bubble tanks in the payload area for the booster orbiter. Both nested and unnested arrangements were evaluated for the propellant requirements. The tank evaluation yielded the need for a stretched version on the booster orbiter. Geometry modifications were performed on the stretched orbiter fuselage to fair the nested double bubble tanks. Mass property evaluations were conducted on the stretched version as well as a separation analysis to determine interference of any booster components.

Space Station Evaluations

This evaluation involved the creation of cornerpoint geometry for a NAR conceived configuration and the EBRO space stations. These configurations included the core module, solar array booms, docking collars, pallets and sub-module. Once the geometry was generated, a compatibility study between the two configurations was conducted. It involved wide use of the EDIN display and geometry manipulation programs. Several configurations were analyzed for compatibility. Discrepancies in the mating of the two systems were identified as well as docking collar modification requirements.

Evaluation of Shuttle with the J2S Engine

This evaluation provided the performance characteristics of a space shuttle vehicle using the J2S engines as the main propulsion system as an interim replacement for the SSME engines. It contained reports on payload characteristics, c.g. and weights analysis and nominal mission trajectory data for a 4 X J2S and

5 X J2S engine configuration. The data was compiled and compared to the SSME configuration. A preliminary sizing study was also conducted on these configurations.

Shuttle Orbiter c.g. Analysis

The EDIN system has been used to determine the controllable c.g. range of the shuttle orbiter through the speed range from Mach 20 down to landing speed. The study combined basic theoretical aerodynamic data with experimental incremental control surface data from the standard aerodynamic data tape. Theoretical aerodynamic data was computed using the DATCOM computer program for the subsonic-supersonic Mach range and HABACP for the hypersonic Mach range. Experimental data was extracted from the standard data tape using the VL70 program.

Sample Analysis

The use of the EDIN system for a design analysis task involves the execution of four major tasks:

1. Define the analysis tasks including the technology areas to be considered, the technological depth and the study parameters.
2. Select the technology programs from the EDIN library and establish the sequence of executions including sizing and matching loops necessary to accomplish the analysis.
3. Establish the data intercommunication requirements of the program including both the fixed input data and the intercommunication data.
4. Generate the run stream required to perform the analysis.

Definition of Analysis Tasks. - Figure 51 illustrates the definition of a group of analysis tasks involving the simple sizing and performance evaluation of a parameter series of all expendable launch vehicles. The study parameters are:

Payload Weight
Number of Stages
Fuel Type
Oxydizer Type
Tank Diameter

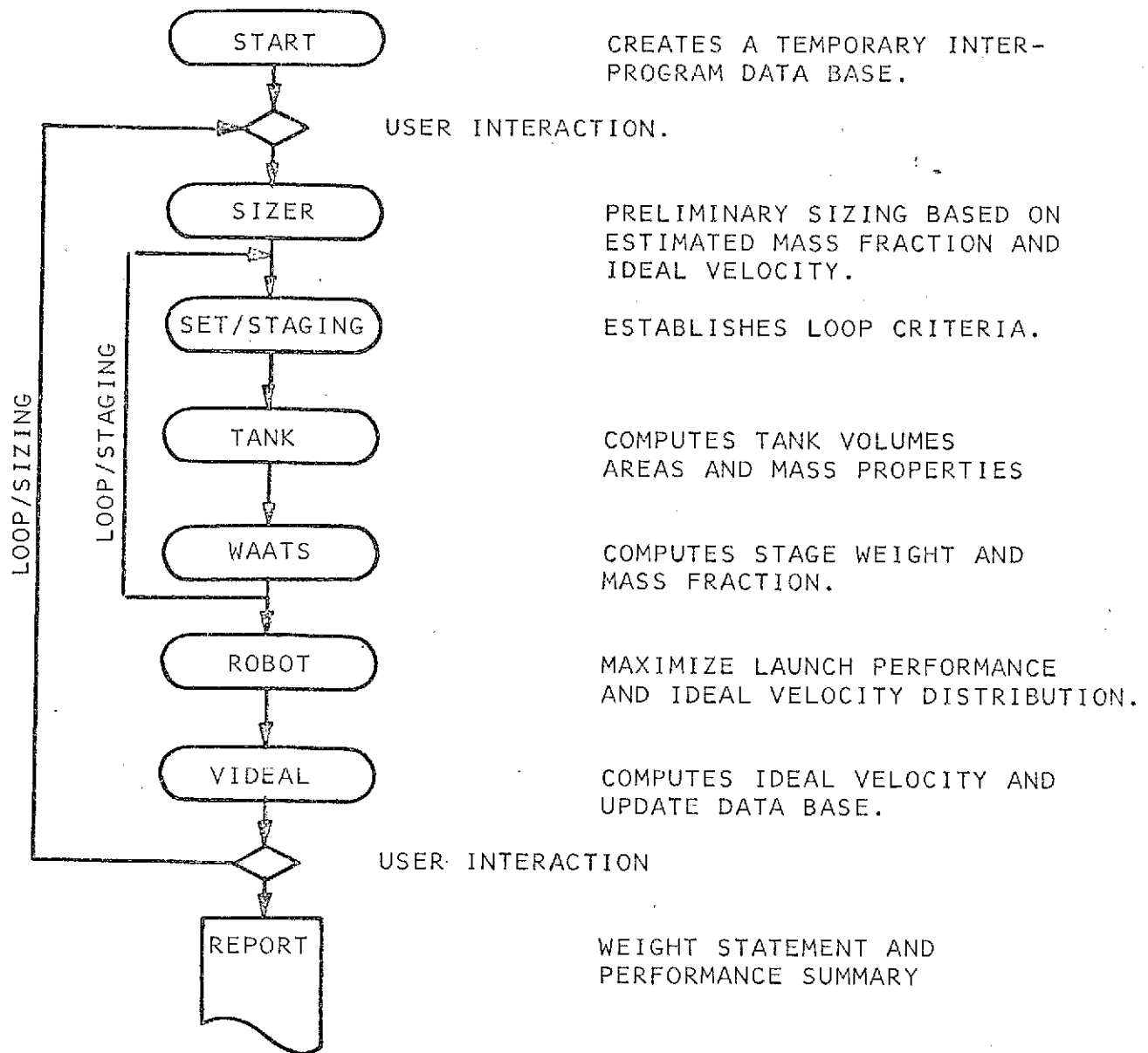


FIGURE 51 LAUNCH PERFORMANCE AND SIZING STUDY.

Thrust-to-Weight Ratio
 Engine Chamber Pressure
 Number of Engines

Selection of Analysis Programs. - The analysis is to be performed using the following EDIN programs:

SIZER	Preliminary Sizing Program which Estimates the Propellant Weights and Inert Weights of all Stages based upon the Input Stage Mass Fraction and Specific Impulse.
TANK	Program to Compute the Fuel and Oxidizer Tank Weights and Geometry based upon the Propellant Quantity, Mixture Ratio, Structural Factors, etc.
WAATS	Weights Analysis Program which Computes all Sub-system Weight based upon Historical Weight Estimating Relations.
ROBOT	Launch Performance Program which Optimizes the Control History and Stage Time.

Intercommunication Data. - The following table describes the program intercommunication requirements for the EDIN programs selected. Fixed input requirements for the program are discussed earlier.

	<u>INPUT</u>	<u>OUTPUT</u>
<u>SIZER.</u> -		
PAYLD - Payload		WSB(I) - Start Burn Weight
NSTAGE - Number of Stages		WEB(I) - End Burn Weight
XLAMDP(I) - Mass Fraction		WPR(I) - Propellant Weight
VIDEAL(I) - Ideal Velocity		WI(I) - Inert Weight
XISP(I) - Specific Impulse		WPL(I) - Stage Payload
WPEST(I) - Propellant Estimate		
<u>TANK.</u> -		
DTANK(I) - Diameter of Tanks		Tank Geometry (File 8)
WPR(I) - Propellant Wt.		LT - Length(S)
		VOL - Volume (s)
		SA - Area(s)
		WP - Propellant Wt(s)
		WTK - Tank Weights
		WAC - Accessory Weights
		WTOT - Total Weights

INPUT

OUTPUT

WAATS. -

ENG(I) - Number of Engines	WCOMP(75) - Dry Weight
THR(I) - Thrust	WCOMP(72) - Gross Weight
WTKG - Tank Weight	WPARM(S) - Mass Fraction
SBODY - Body Area	

ROBOT. -

WSBL - Stage Gross Weights	RBPORT - Stage Report
WPL(I) - Payload	
XISP - Specific Impulse	
THR(I) - Thrust	

VIDEAL. -

RBPORT - Stage Report	VIDEAL - IDEAL Velocity
XISP - Specific Impulse	
NSTAGE - Number of Stages	

Generation of a Run Stream. - Each box in figure 51 represents a sequence of tasks. The executive control statements and data requirements to accomplish the tasks are stored as partial run stream elements in the data base file EDIN-WAATS. The linkage of the elements to perform the analysis is shown in figure 52.

The elements will be preprocessed by the DLG processor as the analysis proceeds to satisfy data base requests contained therein. DLG preprocessing produces executable partial run streams which are merged with the run stream using the @ADD control statement.

The sizing analysis is initiated by the control statement:

@ADD EDIN-WAATS.START

The start task sequence shown below performs the necessary file assignments and constructs a temporary intercommunication data base DBASE.

```
@FREE BK
@ASG,T BK.
@ERS TPF$
@USE D,WORK
@USE E,EDIN-WAATS
@FREE 25
@FREE 14
@ASG,T 25
@ASG,T 14
@USE NMLIST,14
@D,DLG,I REPORT
'CREATE DBASE'
@ADD E.CREATE/EEE
```

ORIGINAL PAGE 15
OF POOR QUALITY

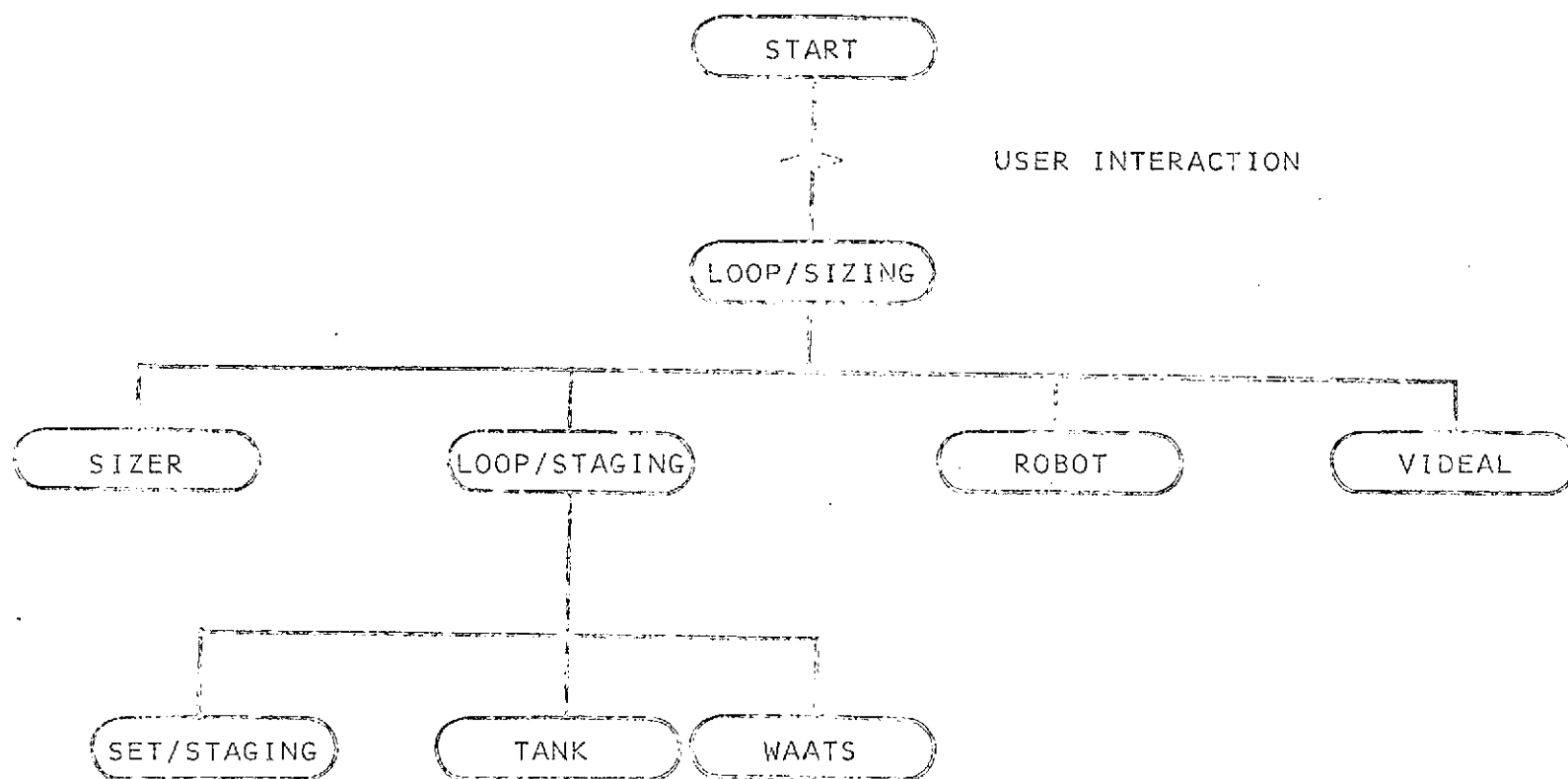


FIGURE 52 PARTIAL RUN STREAM LINKAGE.

BK A temporary output file.

TPF\$ The temporary program file assigned to each run.

WORK The file on which the DLG processor absolute element is stored.

UNIT 25 The interprogram data base.

UNIT 14 The special EDIN output file for EDIN program data.

E A use assigned name for the EDIN-WAATS file.

D A user assigned name for the WORK file.

One page of design data is constructed on Unit 25 through the DLG processor directive:

'CREATE DBASE'

The actual data is stored in the file element EDIN-WAATS.CREATE/EEE as follows:

```
'DEFINE WT=13, MASSP FROM WAB,'
'ADD WT=13*.0,'
      '. OUTPUT FROM TANK,'
'DEFINE LT=2, OX & F TANK LENGTHS,'
'DEFINE VOL=2, OX & F TANK VOLUMES,'
'DEFINE SA=2, OX & F TANK AREAS,'
'DEFINE NP=2, OX & F PROPELLANT WTS,'
'DEFINE WTK=2, OX & F TANK ACC WTS,'
'ADD LT=0.,0.,
      VOL=0.,0.,
      SA=0.,0.,
      NP=0.,0.,
      WTK=0.,0.,
'END
      '. OUTPUT FROM SIZER'
'DEFINE WSB=4, STAGE START BURN WTS,'
'DEFINE WEB=4, STAGE END BURN WTS'
'DEFINE MPL=4, STAGE PAYLOADS,'
'DEFINE WPR=4, STAGE PROP WTS,'
'DEFINE WI=4, STAGE INERT WTS,'
'ADD WSB=0.,0.,0.,0.,
      WEB=0.,0.,0.,0.,
      MPL=0.,0.,0.,0.,
      WPR=0.,0.,0.,0.,
      WI=0.,0.,0.,0.,
'END
      '. OUTPUT FROM WAATS'
'DEFINE WCOMP=77, COMPONENT WTS,'
'DEFINE WPARN=5, NT PARAMETERS,'
'ADD WCOMP=77*.0.,'
```

ORIGINAL PAGE IS
OF POOR QUALITY

The analysis is restarted with the executive control statement:

```
@ADD E.LOOP/SIZING
```

The element LOOP/SIZING shown below contains the control statements and data required to perform the sizing loop shown in figure 51

```
ANUG SIZING LOOP
@D.DLG E.SIZER,SIZER
@ADD SIZER
@BRKPT PRINT$/BK
@BRKPT PRINT$
@D.DLG,I REPORT
  'UPDATE DBASE'
  'PROCESS NMLIST'
  'MSB'
  'WEB'
  'WPR'
  'WI'
@ADD E.LOOP/STAGING
@ADD E.LOOP/STAGING
@ADD E.LOOP/STAGING
@ADD E.LOOP/STAGING
@ENDSTG:
@D.DLG E.ROBOT,ROBOT
@ADD ROBOT
@D.DLG E.VIDEAL,VIDEAL
@ADD VIDEAL
@D.DLG,I0 REPORT
  'UPDATE DBASE'
  'PROCESS NMLIST'
  'ADD S=0'
  'VIDEAL'
  'VLANDP'
@EOF
```

The sizing loop contains instructions for executing the preliminary sizer, the actual weighing of the individual stages, the determination of optimal performance and the calculation of ideal velocity. Preliminary sizing data and control statements are stored in E.SIZER.

ORIGINAL PAGE IS
OF POOR QUALITY

...

```

MSMCH=1,1,
FUELD=1.E6,1.E6,
MSMCH(13)=2,
INDCHI=1,
NOTRAC=0,
KIND=2,
NMAX=20,
STEP=2.,8.,8.,2.,
TZERO=0.,
DTZ=15.,
JORB=1,
NOUT=6,
LAST=0,
HOWD=2,
GLING=5,5,5,5,
KNTA=2,
JTHR=0,
K4010T=11,
$
@EOF

```

The requirements for determining ideal velocity is contained in the element E.VIDEAL.

```

@KOT MAAT82.OUIDEAL
'UPDATE DBASE'
'PROCESS NMLIST'
$IN REPORT='REPORT'
NISP='XISP'
NSTAGE='NSTAGE'
$

```

The staging loop shown below contains the control statements and data required to generate weights data for the number of stages specified. The three major elements are the establishment of the looping criteria, the calculation of tank weight and the calculation of subsystem weights.

```

@HOG STAGING LOOP
@D.DLG E.SET/STAGING,SET
@ADD SET
@D.DLG E.TANK,TANK
@ADD TANK
@D.DLG,I REPORT
'UPDATE DBASE'
'PROCESS NMLIST'
REPORT FROM TANK
LT='LT'
VOL='VOL'

```

ORIGINAL PAGE IS
OF POOR QUALITY

The looping criteria is contained in the element E.SET/STAGING.

When the above element is processed by DLG, the @SETC control statement will contain the stage counter 'S' from the data base. The @TEST control statement will contain the study parameter 'NSTAGE' from the data base. The above control statements, together with the @JUMP control statement, will cause the Exec 8 control to pass to the statement label @ENDSTG when the stage counter 'S' reaches a value of 'NSTAGE.' Since @ENDSTG is in the element E.LOOP/SIZING, the staging loop will be terminated and the launch performance sequence will be started.

[illegible]

1.4
1.4
30.
1
1
0.
0.
1
0.
24.

The control statements and data for the calculation of sub-system weights are contained in the element E.WAATS.

```
'UPDATE DEASE'
@XOT WAATS2.00WAATS
$INMAP
IPRINT=0,
IENG=1,
CREM=0,
'ADD DTI=DTANK(3)'
'ADD ELBODY=LT(1)+LT(2)+DTANK/12.'
ELBODY='ELBODY ',
'ADD TWRI=TNR(3)'
'ADD NSBI=NSB(3)'
'ADD ENGI=ENG(3)'
ENGINS='ENGI',
HBODY='DTI/12. ',
ISHAPE=1,
OF=6,
OFACS=0,
'ADD PCI=PC(3)'
PCHAM='PCI ',
'ADD SA1=SA(1)'
'ADD STPS=SA1+SA(2)/144.'
STPS='STPS ',
'ADD SBODY=STPS*1.25'
SBODY='SBODY ',
SFUTK='SA(2)/144.',
SOKTK='SA1/144.',
TANKS=1,
'ADD TENG=TWRI*NSBI/ENGI'
THRUST='TENG ',
UFUTK=30000,
UOXTK=20000,
'ADD WEBI=WEB(3)'
WLANDI='WEBI ',
'ADD WPLI=WPL(3)'
WPAULD='WPLI '
```

ORIGINAL PAGE IS
OF POOR QUALITY

CONCLUDING REMARKS

A computer aided design environment has been created with the EDIN system. The components of the system are an Univac 1100 series computer and associated software, a flexible data base management system and a library of independent technology computer programs. The EDIN analysis is formulated as a run stream in the language of the computer operating system and the technology module input data. Both elements of the run stream can be augmented with data base requests which are satisfied by a special processor called DLG. The EDIN system provides the users with the ability to formulate the computer aided design problem at the task level in much the same manner as is employed in the industrial design process. The process can be interrupted at any point in the analysis. Demand interaction with the programs and data bases can be performed. Each program is executed sequentially and is "unaware" of its contribution to a larger and more comprehensive engineering process. The result is a flow of program executions which are identical to the normal flow but with a higher degree of control over data intercommunication.

The EDIN system represents a major step towards the development of a true computer aided design environment at Johnson Spacecraft Center.

REFERENCES

1. Gregory, T. J., Peterson, R. J. and Wyss, J. A.: Performance Tradeoffs and Research Problems for Hypersonic Transonic Transports. AIAA Journal of Aircraft. July-August 1965.
2. Gold, R. and Ross, S.: Automated Mission Analysis Using a Parametric Sensitivity Executive Program. AAS Paper 68-146, presented at the AAS/AIAA Astronautics Specialist Conference. September 1968.
3. Wennegal, G. J., Mason, P. W. and Rosenbaum, J. D.: IDEAS, Integrated Design and Analysis System. SAE Paper 68-0728, Presented to SAE Aeronautics and Space Engineering Meeting. October 1968.
4. Adams, J. D.: Vehicle Synthesis of High Speed Aircraft, VSAC, Volume I. USAF AFFDL-TR-71-40. 1971.
5. Oman, B.: Vehicle Synthesis for High Speed Aircraft, VSAC, Volume II. USAF AFFDL-TR-71-40. 1971.
6. Lee, V. A., Ball, H. G., Wadsworth, E. A., Moran, W. J. and McLead, J. D.: Computerized Aircraft Synthesis. AIAA Journal of Aircraft. September-October 1967.
7. Herbst, W. B. and Ross, H.: Application of Computer Aided Design Programs for the Management of Fighter Development Projects. AIAA Paper 70-364, Presented to AIAA Fighter Aircraft Conference. March 1970.
8. Wallace, R. E.: A Computerized System for the Preliminary Design of Commercial Airplanes. AIAA Paper 72-793. Los Angeles, California. 1972.
9. Hague, D. S. and Glatt, C. R.: Optimal Design Integration of Military Flight Vehicles - ODIN/MFV. AFFDL-TR-72-132. 1973.
10. Glatt, C. R., Hague, D. S. and Watson, D. A.: ODINEX: An Executive Computer Program for Linking Independent Programs. NASA CR-2296. National Aeronautics and Space Administration. Washington D. C. September 1973.
11. Fulton, R. E., Sobieszczanski, J. and Landrum, E. A.: An Integrated Computer System for Preliminary Design of Advanced Aircraft. AIAA Paper 72-796. Los Angeles, California. 1972.

12. Rhodes, T. R.: The Computer Aided Design Environment (COM-RADE) Project. Presented at the 1973 National Computer Conference. New York. June 1973.
13. Rau, Timothy R. and Decker, John P.: ODIN: Optimal Design Integration System for Synthesis of Aerospace Vehicles. AIAA Paper No. 74-72. AIAA 12th. Aerospace Sciences Meeting. 1974.
14. Love, Eugene S.: Advanced Technology and the Space Shuttle. Astronautics and Aeronautics, Volume II, No. 2. February 1973.
15. Henderson, Arthur, Jr.: Aerothermodynamic Technology for Space Shuttle and Beyond. AIAA Paper No. 73-59, presented at the AIAA 9th. Annual Meeting and Technical Display. January 1973.
16. Phillips, W. Pelham, Decker, John P., Rau, Timothy R. and Glatt, C. R.: Computer Aided Space Shuttle Orbiter Wing Design Study. NASA TN D-7478. 1973.
17. Harris, R. V., Jr.: An Analysis and Correlation of Aircraft Wave Drag. NASA TM X-947. 1974.
18. Norton, P. and Glatt, C. R.: VAMP: A Computer Program for Calculating the Volume, Area and Mass Properties of Aerospace Vehicles. NASA CR-2419. 1974.
19. Vanco, Michael: Computer Program for Design Point Performance of Turbojet and Turbofan Engine Cycles. NASA TM X-1340. 1967.
20. Fishbach, Laurence H. and Koenig, Robert W.: GENENG II - A Program for Calculating Design and Off-Design Performance of Two and Three Spool Turbofans with as Many as Three Nozzles. NASA TN D-6553. 1972.
21. Wilwerth, R. E., Rosenbaum, R. C. and Chuck, Wong: PRESTO: Program for Rapid Earth to Space Trajectory Optimization. NASA CR-158. 1965.
22. Stein, L. H., Matthews, M. L. and Frenk, J. W.: STOP - A Computer Program for Supersonic Transport Trajectory Optimization. NASA CR-793. 1967.

23. Kinsey, Don W. and Bowers, Douglas L.: A Computerized Procedure to Obtain the Coordinates and Section Characteristics of NACA Designated Airfoils. Technical Report AFFDL-TR-71-87. Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, Ohio. 1971.
24. Hague, D. S. and Glatt, C. R.: An Introduction to Multi-Variable Search Techniques for Parameter Optimization. NASA CR-73200. 1968.
25. Swann, R. T., et. al.: One-Dimensional Numerical Analysis of the Transient Response of Thermal Protection Systems. NASA TN D-2976. 1965.
26. Ayren, H. E.: DAPCA: A Computer Program for Determining Aircraft Development and Production Costs. Rand Corporation Report RM-5221-PR. 1967.
27. Glatt, C. R.: IMAGE: A Computer Code for Generating Picture-Like Images of Aerospace Vehicles. NASA CR-2430. 1974.
28. Glatt, C. R., Hague, D. S. and Reiners, S. J.: Prediction of Sonic Boom from Experimental Near-Field Overpressure Data, Method and Results. NASA CR-2441. 1974.
29. Glatt, C. R., Reiners, S. J. and Hague, D. S.: Prediction of Sonic Boom from Experimental Near-Field Overpressure Data, Data Base Construction. NASA CR-2442. 1974.
30. Glatt, C. R.: WAATS: A Computer Program for Weights Analysis of Advanced Transportation Systems. NASA CR-2420. 1974.
31. Gentry, A.: Hypersonic Arbitrary Body Aerodynamic Program. Douglas Report. DAC56080. June 1967.
32. Brauer, G. L., Cornick, D. E., Steinhoff, R. T. and Stevenson, R.: Program to Optimize Simulated Trajectories (POST). Martin Marietta. NASA MCR-73-206. October 1973.
33. Reiners, S. J., Glatt, C. R., Hirsch, G. N. and Alford, G.: GTM: Geometry Technology Module. Aerophysics Research Corporation. JTN-09. December 1974.
34. Glatt, C. R. and Hirsch, G. N.: PLOTTR: An Independent Computer Program for the Generation of Graphical Displays. Aerophysics Research Corporation. JTN-07. December 1974.

35. Glatt, C. R. and Colquitt, W. N.: The DLG Processor - A Data Management Executive for the Engineering Design Integration System (EDIN). Aerophysics Research Corporation. JTN-10. December 1974.
36. Braley, Dennis M.: Users Guide for the Automatic Flow Chart Generator Program (FLOGEN). JSC Internal Note No. 73-FM-62. April 1973.